sinclair

QL

Benutzer Handbuch

Einführung
Anfängerkurs
Befehle
Begriffe
QL Programme
QL Quill
QL Abacus
QL Archive
QL Easel
Information

HINWEIS AN DEN BETREIBER: WIRD DIESES GERÄT MIT EINEM ODER MEHREREN ANDEREN GERÄT(EN) ZU EINER ANLAGE ZUSAMMENGESCHLOSSEN; SO MUSS DIESE ANLAGE INSGESAMT DEN TECHNISCHEN VORAUSSETZUNGEN VON §2 ZIFFER 1 VFG 1046/1984 (ALLGEMEINE GENEHMIGUNG DER DEUTSCHEN BUNDESPOST NACH DEM HOCHFREQUENZGESETZ) ENTSPRECHEN.

UM FUNKSTÖRUNGEN ZU VERMEIDEN, MUSS BEIM COMPUTERBETRIEB DIE ANTENNENANLAGE VOM FERNSEHGERÄT GETRENNT WERDEN.

ACHTUNG

DAS NETZGERÄT EU2000 DARF NUR MIT DEM COMPUTER SINCLAIR QL BETRIEBEN WERDEN.

QL Handbuch, 2. Ausgabe
Herausgegeben von Sinclair Research Limited 1985
25 Willis Road, Cambridge, England
Erstellt von Stephen Berry (Sinclair Research Limited)

sinclair QL Einführung

WICHTIG

BEVOR SIE DIE APPLIKATIONEN BENUTZEN, LESEN SIE BITTE DIE NACHSTEHENDE BESCHREIBUNG, UM ARBEITSKOPIEN ZU ERSTELLEN.

ARBEITSKOPIEN

Fertigen Sie bitte immer mindestens eine Arbeitskopie an, bevor Sie eines der vier QL Programme benutzen. Arbeiten Sie dann immer nur mit der Arbeitskopie. Bewahren Sie die Originalkassetten mit den Programmen sicher auf. Benutzen Sie die Originalkassetten nur, um bei Bedarf Kopien anzufertigen. So schützen Sie sich vor dem Verlust Ihrer Programme.

Bei allen magnetischen Speichermedien, also auch bei Microdrive-Kassetten, kann bei Gebrauch Datenverlust entstehen. Daher sollten Sie alle Programme und Daten, die Sie noch benötigen, mindestens zweimal speichern. Fertigen Sie sich deshalb immer Sicherungskopien an. So schützen Sie sich vor Datenverlust. Immer, wenn Sie einer Kassette Daten oder Programme hinzufügen, sollten Sie das auch bei der Sicherungskopie tun. Wenn Sie das versäumen, laufen Sie Gefahr, alles, was Sie seit der letzten Datensicherung hinzugefügt haben, zu verlieren. Hinweise zur Anfertigung von Sicherungskopien finden Sie im Abschnitt *Information*.

IHR NEUER QL

Herzlichen Glückwunsch zu Ihrem neuen QL. Damit besitzen Sie einen äußerst leistungsfähigen **Personal Computer**.

Sie werden viel Freude haben, wenn Sie sich die Leistungsfähigkeit Ihres neuen persönlichen Computers erschließen. Dabei will Ihnen dieses Anwenderhandbuch helfen.

Die Möglichkeiten, die Ihnen ein Computersystem wie der QL bietet, sind nicht einfach Seite nach Seite auf bedrucktem Papier zu beschreiben. Die ganze Macht eines **Q**uantum **L**eap – eines Quantensprungs in der Computerentwicklung – wird sich Ihnen aber bald erschließen.

Sie werden erstaunt sein, wie einfach Sie sich Ihren neuen QL dienstbar machen. Er bringt ja vier wichtige Gehilfen gleich zu Ihnen mit – vier ausgeklügelte Arbeitsprogramme, die auf Anhieb für Sie tätig sein können, denn sie sagen Ihnen auf Schritt und Tritt, was Sie tun müssen, damit sie die Arbeit zu Ihrer vollsten Zufriedenheit erledigen. Am Bildschirm, ohne daß Sie die Arbeit unterbrechen müssen.

Auf dem weiteren Weg, die Möglichkeiten Ihres QL auszuschöpfen, hilft Ihnen dieses Handbuch in vielseitiger Form. Wie Sie es dabei am besten benutzen und welche Informationen Sie wo finden, können Sie anschließend lesen.

DER GEBRAUCH DES HANDBUCHES

Je nach dem gegenwärtigen Schwerpunkt Ihres Interesses gibt es zwei verschiedene Wege, sich den QL durch das Handbuch zu erschließen. In jedem Fall sollten Sie zuerst den Abschnitt Einführung durcharbeiten. Dort steht, wie Sie den QL in Betrieb nehmen.

ARBEIT MIT DEN VIER QL PROGRAMMEN

Wenn Sie anfangs vorwiegend mit den vier Programmen zur **Textverarbeitung, Datenverwaltung, Tabellenkalkulation** und **Grafik** arbeiten wollen, so finden Sie im Abschnitt *Einführung* auch eine *Einführung in die QL Programme.* Diese Einführung sagt Ihnen alles über das **Laden der Programme,** wie Sie **grundsätzlich mit ihnen arbeiten** und wie Sie z.B. **mit den Programmen im Netzwerk arbeiten** können.

Jedes Programm ist in einem eigenen Abschnitt dieses Handbuches ausführlich beschrieben. Diese Beschreibungen sind jeweils als Lehrgang aufgebaut, der Ihnen eine Einführung in die Arbeitsweise von Grund auf bietet.

In einem eigenen Abschnitt *INFORMATION zu den QL Programmen* finden Sie dann alles über

Import–Export	den Datenaustausch zwischen den einzelnen Programmen.
Drucker	und die Anpassung Ihres eigenen Druckers an die QL Programme.
Sortierfolge	und ihre Änderung nach Ihren Bedürfnissen.
Laufwerke	und die Anpassung der Programme an geänderte Laufwerke und Einheiten.

DEN QL SELBST PROGRAMMIEREN

Zu diesem Zweck bietet Ihnen der QL ein hervorragendes SuperBASIC, das Möglichkeiten besitzt, die Sie sonst nur in Programmiersprachen finden, die erheblich schwieriger zu erlernen sind. Die dazu nötigen Informationen finden Sie in folgenden drei Abschnitten des Handbuches:

Anfängerkurs
Befehle
Begriffe.

Was Sie im *Anfängerkurs* wo finden, sagt Ihnen das Inhaltsverzeichnis. Die Abschnitte *Befehle* und *Begriffe* sind alphabetisch geordnet. Sie werden darüber hinaus durch ein Register erschlossen.

Wenn Sie erstmalig einen Computer selbst programmieren, ist es sicher vernünftig, Sie arbeiten zuerst den *Anfängerkurs* von Anfang an durch. Haben Sie bereits Erfahrung, besonders mit BASIC, können Sie gleich bei Kapitel 8, *von BASIC zu SuperBASIC,* einsteigen.

Im *Anfängerkurs* sind alle wichtigen Grundlagen der Arbeit mit SuperBASIC eingehend beschrieben. Wenn während der Arbeit Unklarheiten auftreten, können Ihnen die Abschnitte *Befehle* und *Begriffe* sicher weiterhelfen.

Im Abschnitt *Befehle* werden alle SuperBASIC-Befehle genau beschrieben und mit kleinen Beispielen erläutert. Außerdem steht bei jedem Befehl, zu welcher Gruppe von Befehlen er gehört, so daß Sie weitere Informationen über Zusammenhänge im Abschnitt *Begriffe* leicht nachschlagen können.

Der Abschnitt *Begriffe* bietet Ihnen zusammenhängende Informationen: über Möglichkeiten von SuperBASIC z.B. in **Fenstertechnik** und **Grafik**; über die **interne Organisation des** QL und über das **QDOS-Betriebssystem**; über die Arbeit mit **zusätzlichen Peripheriegeräten** u.v.a.m.

Wenn Sie dann viel Erfahrung im Umgang mit Ihrem QL gewonnen haben, wollen Sie vielleicht noch tiefer einsteigen und weitere Möglichkeiten nutzen. Auch dazu stehen Ihnen Hilfen bereit:

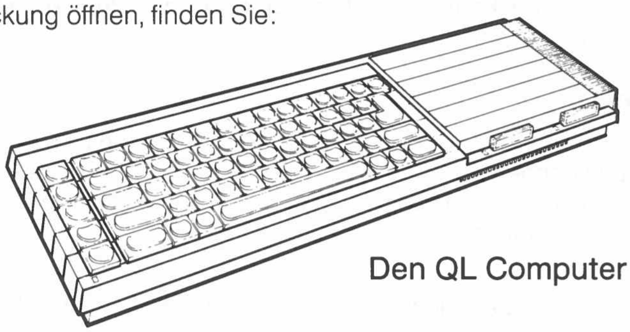
Software Der **QL Tool-Kit** mit vielseitigen Programmierwerkzeugen.
Der **QL Assembler**, mit dem Sie dann auch alle Möglichkeiten des **Multitasking** nutzen können.
Der **QL Monitor**, mit Programmierhilfen zur Entwicklung von Assemblerprogrammen.
Sowie schließlich das Angebot von Programmen durch den Handel, das sich ständig vergrößert.

An **Literatur** empfehlen wir Ihnen dann vor allem das **Technische Handbuch zum QL**, in dem das **QDOS-Betriebssystem** vollständig beschrieben ist und das viele weitere Informationen zur Arbeit mit Ihrem QL bereithält.

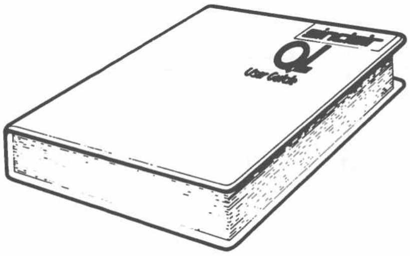
Und nun: Viel Freude beim Entdecken aller Möglichkeiten Ihres neuen QL.

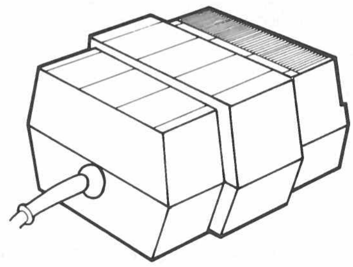
EINFÜHRUNG IN DEN QL

Wenn Sie die Verpackung öffnen, finden Sie:



Den QL Computer





Das QL Anwenderhandbuch

Das Netzteil

Zwei Taschen mit Microdrive-Kassetten

Eine enthält:

- QL Abacus
- QL Archive
- QL Easel
- QL Quill

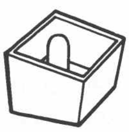


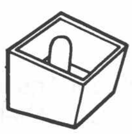


Die andere enthält:

- QL Utilities
- 3 leere Microdrive-Kassetten

Zwei Kunststoffüße





Mit Ihnen können Sie den QL in eine schreibgerechte Lage bringen.

Stecken Sie die Nasen der Stifte in die Löcher der hinteren Gummifüße an der Unterseite des QL. Durch leichtes Drehen werden sie befestigt.

Ein Antennenkabel

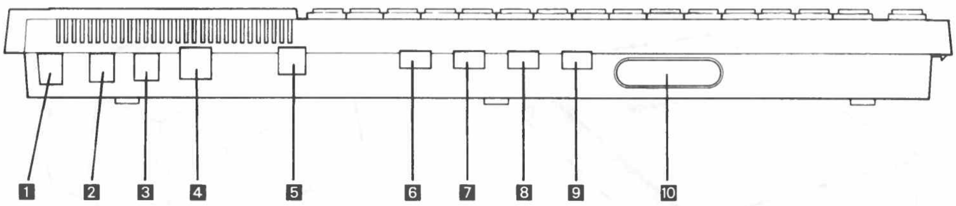
Es ist ungefähr zwei Meter lang und hat an beiden Enden verschiedenartige Stecker. Mit diesem Kabel können Sie Ihren QL mit dem Antennenanschluß Ihres Fernsehgerätes verbinden.

Ein Netzwerkkabel

Es ist auch rund zwei Meter lang. Die Stecker sind an beiden Enden gleich. Mit diesem Kabel können Sie Ihren QL mit anderen QL's verbinden. Damit können Sie ein Netzwerk aufbauen, in dem es möglich ist, Daten und Programme zwischen mehreren QL's auszutauschen.

ANSCHLÜSSE

Am Computer sehen Sie hinten und an beiden Seiten eine Reihe von Anschlüssen.

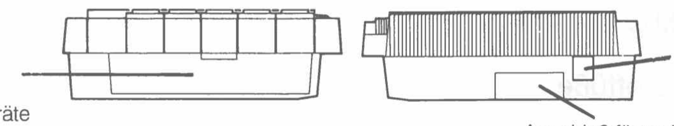


Die Anschlüsse dienen dazu, Verbindungen zwischen dem QL und folgenden **Zusatzgeräten** herzustellen:

1	**NET**	QL Netzwerk	E/A
2	**NET**	QL Netzwerk	E/A
3	**POWER**	Stromversorgung	–
4	**RGB**	Farb- oder Monochrommonitor	A
5	**UHF**	Fernsehgerät	A
6	**SER1**	serielle Datenübertragung RS-232C	E/A
7	**SER2**	serielle Datenübertragung RS-232C	E/A
8	**CTL1**	Joystick	E
9	**CTL2**	Joystick	E
10	**ROM**	Software auf QL ROM Kassetten	E

E: Der Anschluß dient nur zur Eingabe.
A: Der Anschluß dient nur zur Ausgabe.
E/A: Der Anschluß dient gleichermaßen zur Eingabe wie zur Ausgabe.

ZX ROM Kassetten können nicht mit dem QL benutzt werden!



An der linken Seite befindet sich ein Anschluß, über den Sie den QL mit weiteren Zusatzgeräten verbinden können. Damit haben Sie die Möglichkeit, die Leistungsfähigkeit Ihres QL noch erheblich zu erweitern. Dieser Anschluß ist mit einer Kunststoffkappe abgedeckt.

Die Rücksetztaste befindet sich an der rechten Seite des QL. Mit ihr können Sie jederzeit den QL praktisch in den Zustand zurücksetzen, den er unmittelbar nach dem Einschalten hatte. Beachten Sie aber bitte: Alle Programme und Daten, die im QL gespeichert sind, gehen dabei verloren. Benutzen Sie deshalb die Rücksetztaste immer mit Vorsicht.

Ebenfalls an der rechten Seite befindet sich ein Anschluß, über den Sie den QL mit bis zu sechs weiteren **Microdrives** verbinden können. Dieser Anschluß ist mit einer Kunststoffkappe abgedeckt.

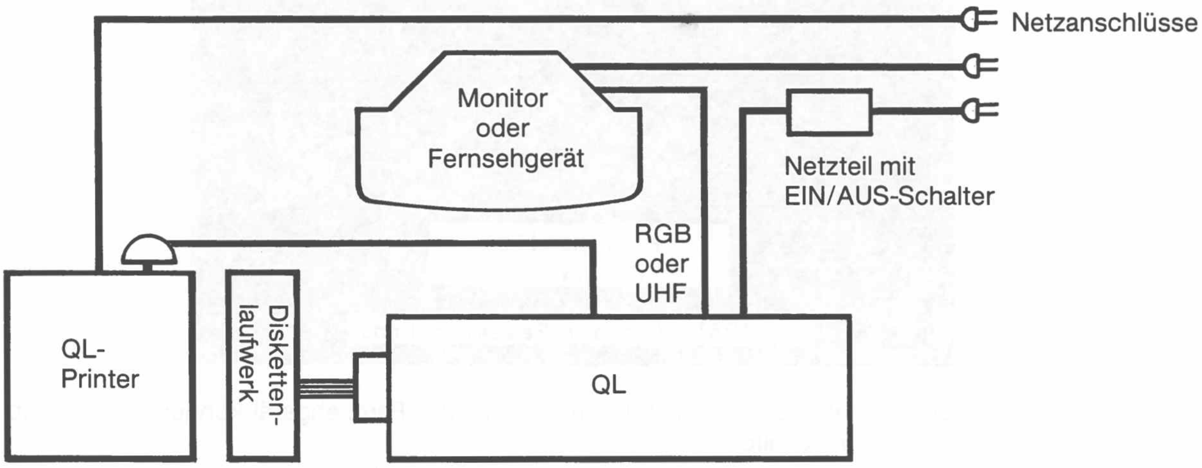
ANSCHLIESSEN

Das Netzteil

Schon mit zwei Verbindungen ist Ihr QL betriebsbereit.

Das Netzteil besitzt zwei Kabel. Eines wird mit der Steckdose verbunden. Der kleine Stecker des anderen Kabels wird in den Anschluß an der Rückseite des QL gesteckt, der mit POWER bezeichnet ist. Am Netzteil befindet sich auch der Ein/Ausschalter.

Wenn das Netzteil angeschlossen und der Schalter eingeschaltet ist, dann leuchtet die kleine gelbe Lampe links vorne an der Tastatur: Der QL ist in Betrieb.



Bitte beachten Sie grundsätzlich: Wenn Sie irgendwelche Zusatzgeräte anschließen, müssen Sie den Computer immer ausschalten. Schalten Sie ihn erst dann wieder ein, wenn alle Geräte, die Sie zum Betrieb benötigen, richtig angeschlossen sind. Andernfalls könnte der Computer oder das Zusatzgerät beschädigt werden.

Der Bildschirm

Wenn Sie den Netzanschluß hergestellt haben, arbeitet der QL. Nur zeigt er Ihnen noch nicht, was er tut. Sie benötigen ein Bildschirmgerät, damit Sie es sehen können.

Der QL Monitor

Ein Monitor sieht ähnlich aus, wie ein Fernsehgerät. Er kann aber kein Fernsehprogramm empfangen. Dafür ist die Anzeige eines Monitors erheblich schärfer und bei längerer Arbeit mit dem Computer weniger ermüdend für Ihre Augen.

Der QL Monitor besitzt zwei Kabel. Das Kabel mit dem kleinen runden Stecker wird in den RGB Anschluß an der Rückseite des QL gesteckt. Das andere Kabel wird mit einer Netzsteckdose verbunden.

Mit dem QL Monitor können Sie alle Möglichkeiten der farbigen Ausgabe von Ergebnissen Ihres Computers nutzen.

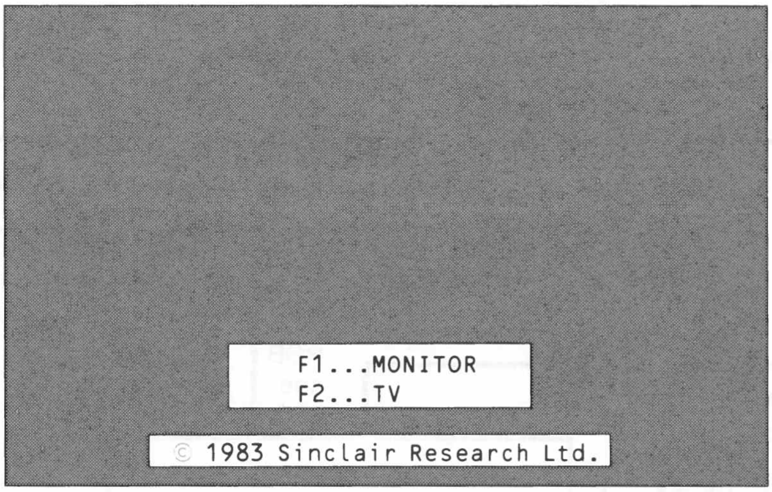
Wenn Sie einen anderen Monitor benutzen wollen, benötigen Sie ein spezielles Verbindungskabel. Auf der einen Seite muß es mit einem normalen achtpoligen DIN-Stecker versehen sein, der in den RGB-Anschluß des QL paßt. Der Stecker, der an der anderen Seite benötigt wird, ist von Fabrikat zu Fabrikat verschieden. Deshalb ist es nicht möglich, ein Kabel zu liefern, das an alle handelsüblichen Monitore paßt. Einzelheiten über den Anschluß anderer Monitore an den QL finden Sie im Abschnitt *Begriffe* unter *Monitor*.

Das Fernsehgerät

Sie können den QL auch mit praktisch jedem üblichen Fernsehgerät verbinden. Der QL sendet das Signal im gleichen Bereich, in dem Sie das ZDF empfangen können.

Die beiden Stecker des mitgelieferten Antennenkabels sind verschieden. Stecken Sie den Stecker, der dem Ihrer Fernsehantenne ähnlich sieht, in die Antennenbuchse Ihres Fernsehgerätes. Wenn Ihr Fernsehgerät getrennte Antennenbuchsen für VHF und UHF besitzt, müssen Sie das Kabel über einen Zwischenstecker mit der UHF-Buchse verbinden.

Jetzt wird der QL eingeschaltet und das Fernsehgerät auf das QL Signal abgestimmt. Das QL Signal liegt bei Kanal 36. Stimmen Sie den Fernseher also in einem Bereich ab, in dem Sie auch das ZDF empfangen können. Wenn Sie Ihr Gerät richtig abgestimmt haben, dann zeigt es folgendes Bild, vorausgesetzt, Sie haben keine Tasten am QL gedrückt.



Der QL hat einen Lautsprecher eingebaut. Am Fernsehgerät können Sie deshalb den Ton abschalten.

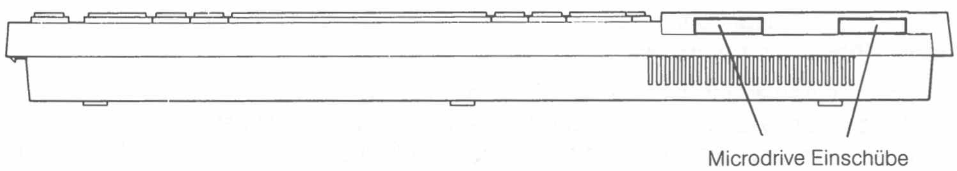
Sollten Sie dieses Bild nicht bekommen, dann prüfen Sie bitte, ob Sie die üblichen Fernsehsender empfangen können. Wenn nicht, versuchen Sie es bitte mit einem anderen Gerät.

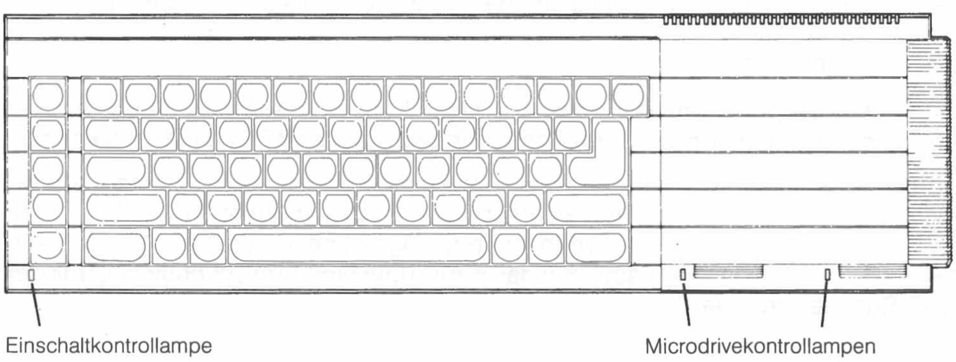
Sollte das Bild verschwommen sein, könnte es an einer fehlerhaften Abstimmung liegen. Möglicherweise liefert der QL bei verschiedenen Abstimmungen ein Bild. Versuchen Sie bitte den ganzen Bereich durchzustimmen, bis Sie ein scharfes Bild empfangen. Überzeugen Sie sich aber vorher, daß das Antennenkabel am QL und am Fernseher richtig eingesteckt ist.

Wenn Sie einen Monochrommonitor oder ein Schwarz/Weiß-Fernsehgerät anschließen, zeigt der QL statt verschiedener Farben verschiedene Grautöne an.

ARBEITEN

Jetzt ist Ihr QL betriebsbereit. Die Bildschirmanzeige sagt Ihnen, daß Sie die Taste **F1** (in der linken Tastenreihe) drücken müssen, wenn Sie einen Monitor angeschlossen haben, andernfalls die Taste **F2**.





Wenn Sie eine der beiden Tasten gedrückt haben, leuchtet die Kontrollampe von **Microdrive 1** kurz auf. Der QL sucht dann nämlich nach einem Programm, das er eventuell von **Microdrive 1** laden kann.

Wenn Sie keine Programm-Kassette eingelegt haben, sehen Sie nach kurzer Zeit ein rotes Blinkzeichen links unten am Bildschirm. Das ist der sogenannte *Cursor*. Jetzt ist der QL bereit, Ihre Aufträge entgegenzunehmen.

DIE TASTATUR

Die Tastatur des QL sieht fast aus wie eine normale Schreibmaschinentastatur. Sie werden nur einige Tasten finden, die auf der Schreibmaschine nicht vorhanden sind. Zuerst soll die Bedeutung dieser Tasten erläutert werden.

Enter

Die mit ↵ gekennzeichnete Taste nennen wir **ENTER**. Sie dient dazu, dem Computer mitzuteilen, daß die Eingabe eines Wertes oder eines Befehls beendet ist, und sie veranlaßt ihn, die Verarbeitung zu beginnen.

Shift

Die beiden mit ⇧ gekennzeichneten Tasten nennen wir **SHIFT**. Sie dienen dazu, Großbuchstaben auszugeben. Bei anderen als Buchstabentasten wird jeweils das Zeichen ausgegeben, das oben auf der Taste steht.

⇧ mit **5** gibt % aus.

Umschaltung

Wenn Sie die *Umschalttaste* (im englischen **CAPS LOCK** ⇧) einmal betätigen, werden Großbuchstaben ausgegeben. Bei anderen als den Buchstabentasten hat die *Umschalttaste* keine Wirkung. Die Wirkung der Umschaltung bleibt erhalten, bis die Taste erneut betätigt wird.

Löschen

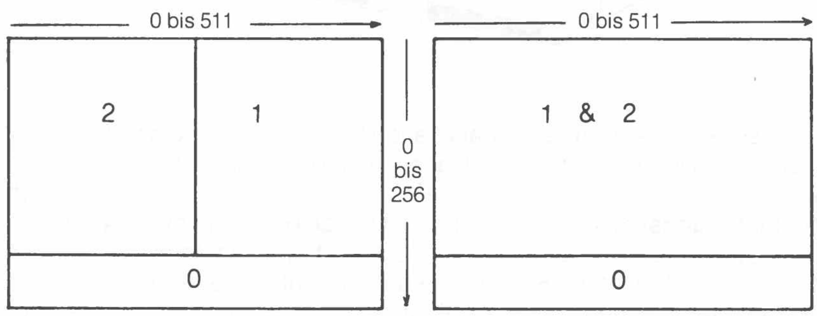
Geben Sie einige Buchstaben über die Tastatur ein. Halten Sie die **CTRL** Taste gedrückt und drücken Sie dann auf die [←] Taste. Das Zeichen, auf dem der Cursor steht, verschwindet. Der *Cursor* rückt dann um ein Zeichen nach links. Bewegen Sie jetzt mit der [←] Taste den Cursor um einige Zeichen nach links. Halten Sie anschließend wieder die **CTRL** Taste gedrückt und drücken Sie auf die [→] Taste. Jetzt verschwindet das Zeichen rechts vom *Cursor*.

Funktionstasten

Die Tasten der linken Reihe sind mit **F1** bis **F5** beschriftet. Es sind *Funktionstasten*. Mit ihnen können Sie bei den Programmen bestimmte Funktionen auswählen. Wenn immer Sie im Handbuch oder am Bildschirm aufgefordert werden, eine *Funktionstaste* zu betätigen, so erreichen Sie den gewünschten Erfolg durch Betätigen einer einzigen Taste aus der linken Reihe und nicht etwa durch Betätigen der Taste **F** gefolgt von einer Zifferntaste.

DER BILDSCHIRM

Wenn Sie nach dem Einschalten oder Rücksetzen die *(Funktions-)Taste* **F1** oder **F2** betätigt haben, dann zeigt der Bildschirm eines der beiden folgenden Bilder.



Monitor   Fernsehgerät

Der Bildschirm ist in verschiedene Bereiche unterteilt, die wir **Fenster** nennen. Das schmale lange **Fenster** am unteren Bildschirmrand wird benutzt, um Befehle anzuzeigen, die in den Computer eingegeben werden. Zu Beginn zeigt es den blinkenden *Cursor*. Wenn der *Cursor* sichtbar ist, dann ist der QL bereit, Befehle anzunehmen. Der *Cursor* verschwindet, wenn der QL beschäftigt ist. Immer wenn Sie ein Zeichen eingeben, wandert der *Cursor* um eine Stelle nach rechts.

Wenn der QL einmal nicht richtig reagiert, können Sie ihn unterbrechen. Halten Sie dazu die **CTRL** Taste gedrückt und betätigen Sie die Leertaste. Auf diese Weise können Sie auch ein SuperBASIC Programm unterbrechen.

Anschließend sollte am Eingabefenster die Meldung ABGEBROCHEN erscheinen und der Cursor wieder blinken.

Ist das nicht der Fall, dann entfernen Sie bitte eingelegte Microdrive-Kassetten und drücken Sie die Rücksetztaste.

Wenn Sie etwas eingeben, das der QL nicht versteht, dann erscheint die Meldung SYNTAX FEHLER im Eingabefenster. Korrigieren Sie die Eingabe oder unterbrechen Sie, wie oben beschrieben.

MICRODRIVES

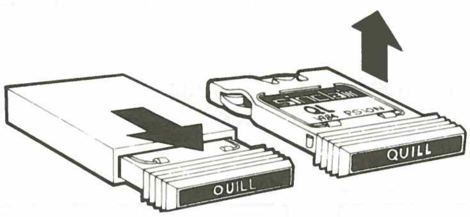
Rechts an der Vorderseite des QL sehen Sie zwei rechteckige Öffnungen. Dort befinden sich die beiden QL **Microdrives,** wovon die linke **mdv1_** und die rechte **mdv2_** ist. Die Kassetten dieser **Microdrives** dienen zum dauerhaften Speichern von Programmen und Daten. Neben jeder Öffnung sehen Sie eine rote Lampe. Wenn sie leuchtet, ist der **Microdrive** in Betrieb. In dieser Zeit dürfen Sie die Kassette nicht herausnehmen.

Setzen Sie die Kassetten immer richtig in die Laufwerke ein. Fassen Sie sie nur an der geriffelten Grifffläche an, und ziehen Sie sie aus der Schutzhülle. Führen Sie sie so ein, daß das Etikett oder die Aussparung für das Etikett nach oben zeigt.

Behandeln Sie die Kassetten bitte immer sorgfältig. Schalten Sie den QL nie ein oder aus und drücken Sie nie die Rücksetztaste, solange sich eine Kassette im Laufwerk befindet. Führen Sie die Kassetten immer vorsichtig ein und nehmen Sie sie vorsichtig heraus. Warten Sie damit immer bis die rote Anzeigelampe erloschen ist. Berühren Sie das Band nie mit den Fingern und stecken Sie die Kassetten nach Gebrauch immer in ihre Schutzhüllen.

Bevor eine Kassette benutzt werden kann, muß sie **formatiert** werden. Wie das gemacht wird, können Sie im Abschnitt *Information* lesen. Aber **Vorsicht**: Alle Programme und Daten, die sich eventuell auf der Kassette befinden, werden beim **Formatieren** gelöscht. Kennzeichnen Sie Ihre Kassetten deshalb bitte immer mit den beigefügten Klebeetiketten.

Sie können Kassetten auch gegen versehentliches Überschreiben schützen. Brechen Sie dazu die **Schreibschutzlasche** an der rechten Seite der Kassette heraus.



Bei allen magnetischen Speichermedien, also auch bei **Microdrive-Kassetten,** kann beim Gebrauch Datenverlust entstehen. Daher sollten Sie alle Programme und Daten, die Sie noch benötigen, mindestens zweimal speichern. Fertigen Sie sich deshalb immer **Sicherungskopien** an. So schützen Sie sich vor Datenverlust. Immer, wenn Sie einer Kassette Daten oder Programme hinzufügen, sollten Sie das auch bei der Sicherungskopie tun. Wenn Sie das versäumen, laufen Sie Gefahr, alles, was Sie seit der letzten Datensicherung hinzugefügt haben, zu verlieren. Hinweise zur Anfertigung von Sicherungskopien finden Sie im Abschnitt *Information.*

WEITERE ZUSATZGERÄTE

QL Printer

Ein *Drucker* ist für Sie, wenn Sie den QL intensiv nutzen wollen, ein fast unentbehrliches Zubehör. Den **QL Printer** schließen Sie einfach an, indem Sie zuerst den breiten Stecker des Verbindungskabels in die passende Buchse an der Rückseite des Druckers stecken. Den kleinen Stecker des Verbindungskabels stecken Sie in den Anschluß **SER1** an der Rückseite des QL. Jetzt müssen Sie noch den Netzstecker des Druckers in die Steckdose stecken sowie den Drucker und den QL einschalten.

Diskettenlaufwerke

Wenn Sie Ihren QL mit einem *Diskettenlaufwerk* betreiben wollen, dann stecken Sie das **Interface** in den Erweiterungsanschluß an der linken Seite des QL ein. Vorher müssen Sie natürlich die Kunststoffabdeckung entfernen.

Das Verbindungskabel des Laufwerkes zum QL wird in den entsprechenden Anschluß des **Interface** eingesteckt. Alles weitere, insbesondere, wie Sie das Laufwerk benennen müssen, entnehmen Sie bitte der *Anleitung für das Diskettenlaufwerk.*

Speichererweiterung

Wenn Sie mit den **QL Programmen** umfangreiche Aufgaben bewältigen müssen oder andere Programme mit großen Datenbeständen einsetzen wollen, dann können Sie den Arbeitsspeicher des QL, der 128 **kilo**B**ytes (kB)** beträgt, um bis zu 512 kB erweitern. Die **Speichererweiterung** wird in den Erweiterungsanschluß an der linken Seite des QL gesteckt. Vorher müssen Sie die Kunststoffabdeckung entfernen.

DIE NÄCHSTEN SCHRITTE

Es gibt viele Möglichkeiten, den QL und dieses Handbuch zu benutzen. Sie können mit den vier Programmen arbeiten, die mit dem QL geliefert werden oder mit anderen fertigen Programmen, die beim Handel zu kaufen sind. Oder Sie schreiben Ihre eigenen Programme.

Bevor Sie mit den vier **QL Programmen** arbeiten, lesen Sie bitte die *Einführung in die QL Programme* auf den folgenden Seiten.

Für die Programmierung in SuperBASIC finden Sie alle nötigen Informationen in den Abschnitten *Anfängerkurs, Befehle* und *Begriffe* in diesem Handbuch. Wenn Sie schon Erfahrung in der BASIC-Programmierung besitzen, können Sie den *Anfängerkurs* gleich bei Kapitel 8 – *von BASIC zu SuperBASIC* – beginnen. Dort werden die wesentlichen Unterschiede zwischen bisherigen BASIC-Versionen und SuperBASIC beschrieben. Wenn Sie sich sicher fühlen, dann können Sie auch gleich mit den Abschnitten *Befehle* und *Begriffe* arbeiten. Eine Übersicht, welche Befehle und Funktionen in SuperBASIC verfügbar sind, finden Sie im Abschnitt *Begriffe* unter dem Stichwort *Befehle.*

WENN PROBLEME AUFTRETEN

Sollten Sie einmal Schwierigkeiten bei der Arbeit mit demQL oder den **QL Programmen** haben, dann empfehlen wir:

1. Versuchen Sie mit diesem Handbuch herauszufinden, wie Ihr Problem zu lösen ist.
2. Ziehen Sie andere Bücher über den QL zu Rate.
3. Wenn Ihr Problem so nicht gelöst werden kann und wenn Sie glauben, daß es durch einen Fehler im QL oder in einer Programm-Kassette verursacht ist, dann lesen Sie bitte am Ende dieses Handbuches über die Garantieleistungen nach.

EINFÜHRUNG IN DIE QL PROGRAMME

Diese Einführung beschreibt kurz die vier Programme, die mit dem QL geliefert werden und zeigt Ihnen Gemeinsamkeiten bei ihrer Anwendung.

Die vier Programme sind:

QL Quill – die Textverarbeitung
QL Abacus – die Tabellenkalkulation
QL Archive – die Datenverwaltung
QL Easel – das Grafikprogramm

Jedes Programm wird in einem eigenen Abschnitt dieses Handbuches ausführlich beschrieben. Lesen Sie diese Abschnitte aber bitte nicht einfach durch. Probieren Sie vielmehr die Beispiele aus und experimentieren Sie mit jedem Vorschlag.

ARBEITSKOPIEN

Fertigen Sie bitte immer mindestens eine **Arbeitskopie** an, bevor Sie eines der vier **QL Programme** benutzen. Arbeiten Sie dann immer nur mit der **Arbeitskopie.** Bewahren Sie die Originalkassetten mit den Programmen sicher auf. Benutzen Sie die Originalkassetten nur, um bei Bedarf Kopien anzufertigen. So schützen Sie sich vor dem Verlust Ihrer Programme.

Jede Programm-Kassette enthält ein Programm, mit dem Sie den Inhalt der Kassette kopieren können. Es wird wie folgt benutzt:

- Legen Sie die Originalkassette in Microdrive 2 ein.
- Legen Sie eine leere Kassette, oder eine Kassette die nichts enthält, was Sie noch benötigen, in Microdrive 1 ein. Geben Sie ein:

  ```
  lrun mdv2_clone
  ```

- Drücken Sie die Taste **ENTER.** Nach kurzer Pause erscheint auf dem Bildschirm die folgende Nachricht:

  ```
  Formatieren mdv1_
  ```

  – weiter mit [LEERTASTE]

- Wenn Sie sicher sind, daß die Kassette in Microdrive 1 nichts enthält, was Sie noch benötigen, können Sie durch Druck auf die Leertaste das Programm fortsetzen. Beim Formatieren wird alles gelöscht, was eventuell auf der Kassette gespeichert ist. Dann wird der Inhalt der Programm-Kassette vom Microdrive 2 auf die Kassette in Microdrive 1 kopiert. Auf dem Bildschirm wird angezeigt, welcher Teil jeweils gerade kopiert wird.
- Warten Sie, bis die Anzeigelampen der Microdrives erloschen sind. Dann können Sie die Programm-Kassette aus Microdrive 2 entfernen.

LADEN

Benutzen Sie die Original-Programm-Kassette höchstens, um eine **Arbeitskopie** anzufertigen. Laden Sie immer nur von der Arbeitskopie.

Alle vier **QL Programme** werden gleichartig geladen. Dazu gibt es zwei Möglichkeiten:

Entfernen Sie eine eventuell eingelegte Microdrive-Kassette. Drücken Sie dann die Rückstelltaste an der rechten Seite des QL. Legen Sie Ihre **Arbeitskopie** des Programms, das Sie benutzen wollen, in Microdrive 1 ein. Drücken Sie die Taste **F1** oder **F2** wie angezeigt. Microdrive 1 läuft dann von selbst an. Nach kurzer Pause sehen Sie ein Titelbild auf dem Bildschirm, das Ihnen anzeigt, welches Programm geladen wird. Wenn der Ladevorgang beendet ist, startet das Programm von selbst.

Wenn Sie mit den Programmen vertrauter sind und etwa einen **Drucker** oder das **Netzwerk** benutzen, kann es nötig sein, daß Sie vor dem Start eines Programms einige Anweisungen in den Computer eingeben müssen. Diese Eingaben gingen verloren, wenn Sie ein Programm dann, wie oben beschrieben, nach Betätigen der Rückstelltaste starten würden. In diesem Fall gehen Sie so vor: Legen Sie die Arbeitskopie des Programms in Microdrive 1 ein. Geben Sie ein

```
lrun mdv1_boot
```

und drücken Sie **ENTER**. Dadurch wird der Ladevorgang in Gang gesetzt. Der weitere Ablauf ist gleich wie oben beschrieben.

In beiden Fällen wird das Programm möglicherweise weitere Informationen laden müssen. Lassen Sie die Kassette deshalb bitte immer eingelegt, bis Sie die Arbeit mit dem Programm beendet haben.

Andere Drucker statt QL Printer

Wenn Sie einen anderen Drucker als den **QL Printer** angeschlossen haben, dann stellen Sie den Drucker auf den deutschen Zeichensatz ein. Auch dem QL müssen Sie dies mitteilen. Dazu gehen Sie wie folgt vor.

Schalten Sie den QL ein und legen Sie keine Microdrive-Kassette ein. Drücken Sie die Taste **F1** oder **F2**. Legen Sie dann die Kassette des Programms, mit dem Sie arbeiten wollen, in Microdrive 1 ein. Geben Sie über die Tastatur ein:

```
load mdv1_boot
list
2 TRA 1
```

jeweils gefolgt von **ENTER**. Die Zeile 2 des gelisteten Programms muß jetzt in **TRA 1** geändert sein. Geben Sie dann noch ein:

```
delete mdv1_boot
save mdv1_boot.
```

Jetzt wird jedesmal, wenn das Programm geladen wird, der QL auf Ihren Drucker eingestellt.

Gehen Sie bei den anderen Programmen, mit denen Sie mit einem Drucker arbeiten wollen, ebenso vor.

Ausnahme: QL Easel; bei diesem Programm muß in Zeile 2 die Anweisung **TRA 0** stehen bleiben.

BILDSCHIRM-DARSTELLUNG

Der eingerahmte **Steuerbereich** im oberen Teil des Bildschirms dient Ihnen als Führer durch die Programme. Dort sehen Sie, welche Programmfunktionen Sie wählen können. Bei Bedarf finden Sie darin oft auch weitere Informationen. Wenn das Programm irgendwelche Eingaben benötigt, macht es Ihnen auch häufig geeignete Vorschläge. Durch Drücken von **ENTER** können Sie den vom Programm vorgegebenen Eingabevorschlag auswählen. Wenn Sie etwas anderes eingeben wollen, als das Programm vorschlägt, geben Sie einfach Ihren eigenen Text ein. Der ursprüngliche Vorschlag verschwindet dann.

Wenn Sie die Taste **F2** drücken, verschwindet der Steuerbereich und der **Anzeigebereich** wird größer. Erneutes Drücken von **F2** läßt den Steuerbereich wieder oben auf dem Bildschirm erscheinen.

Der Anzeigebereich in der Mitte des Bildschirms zeigt Ihnen, was Sie gerade bearbeiten. Z. B. den Text eines Dokumentes bei der Arbeit mit **QL Quill**, den Inhalt einer "Karteikarte" bei der Arbeit mit **QL Archive**, eine Grafik von **QL Easel** oder ein Kalkulationsblatt von **QL Abacus.** Die Darstellung erfolgt in einer Form, die der jeweiligen Aufgabenstellung besonders angepaßt ist.

Der untere Teil des Bildschirms ist der **Arbeits-** oder **Statusbereich.** Dort sehen Sie z. B., welches Kommando Sie gerade eingegeben haben. Außer in **QL Archive** wird dort auch noch einiges über den augenblicklichen Stand der Arbeit angezeigt. Z. B. der Name der Datei oder des Dokumentes das Sie gerade bearbeiten, der verfügbare freie Speicher u. a.

FUNKTIONS-TASTEN

Drei der fünf Funktionstasten am linken Rand der Tastatur haben in allen vier QL Programmen jeweils dieselbe Wirkung:

Taste	Wirkung
F1	Anforderung von Hilfetexten
F2	Löschen oder Anzeigen des Steuerbereiches
F3	Aufruf des Menüs zur Befehlsauswahl

Die beiden anderen Funktionstasten **F4** und **F5** haben in jedem Programm eine eigene Bedeutung.

Bitte beachten Sie: Wann immer Sie in diesem Handbuch oder am Bildschirm aufgefordert werden, eine Funktionstaste zu drücken, dann erreichen Sie den gewünschten Erfolg durch Betätigen einer einzigen Taste in der linken Reihe und nicht etwa durch Betätigen der Taste **F** gefolgt von einer Zifferntaste.

HILFE

Links oben im Steuerbereich sehen Sie, daß Sie mit der Taste **F1 Hilfe** (-texte) aufrufen können.

Wenn Sie **HILFE** anfordern, sehen Sie nach kurzer Pause auf dem Bildschirm Informationen, die Ihnen bei der Arbeit weiterhelfen können.

So z.B. eine Liste von Begriffen, zu denen die **HILFE** weitere Informationen liefern kann. Geben Sie den Namen des Begriffs ein und drücken Sie **ENTER. Sie müssen meist nicht den vollständigen Begriff eintippen. Es reicht, wenn Sie soviele Buchstaben eingeben, daß der gewünschte Begriff eindeutig gekennzeichnet ist. Diesen Vorgang können Sie wiederholen, sooft es nötig ist.**

Wenn Sie **ENTER** drücken, ohne einen Begriff zu wählen, schaltet **HILFE** auf die letzte Darstellungsebene zurück. Mit **ESC** kommen Sie unmittelbar ins Programm zurück. Sie können dann Ihre Arbeit so fortsetzen, als hätten Sie die **HILFE** nicht angefordert.

Hilfe kann immer angefordert werden, solange die Programm-Kassette im Microdrive 1 ist. Drücken Sie **F1**, und die zum jeweiligen Arbeitsstadium passende Information der **HILFE** wird ausgegeben.

DER ZEILENEDITOR

Wenn Sie eine Eingabezeile korrigieren wollen, können Sie den Zeileneditor benutzen. Alle **vier QL Programme** benutzen denselben **Zeileneditor.** Jedes Programm benutzt ihn aber in einer Weise, die seinen Aufgabenstellungen besonders angemessen ist. In **QL Quill** bearbeiten Sie mit dem Zeileneditor z.B. den Text von Kommandos. In **QL Archive** benutzen Sie den **Zeileneditor** ausgiebig bei der Bearbeitung von Datenverwaltungsprogrammen.

Der Zeileneditor wird mit den vier **Pfeiltasten,** der **CTRL** und den **SHIFT**-Tasten gesteuert.

Tasten	Wirkung
←	Setzt den Cursor um eine Stelle nach links
→	Setzt den Cursor um eine Stelle nach rechts
SHIFT mit ←	Setzt den Cursor um ein Wort nach links
SHIFT mit →	Setzt den Cursor um ein Wort nach rechts
CTRL mit ←	Löscht das Zeichen links vom Cursor
CTRL mit →	Löscht das Zeichen, auf dem der Cursor steht
CTRL mit Pfeil auf	Löscht die Zeile links vom Cursor
CTRL mit Pfeil ab	Löscht die Zeile rechts vom Cursor
SHIFT mit CTRL mit ←	Löscht das Wort links vom Cursor
SHIFT mit CTRL mit →	Löscht das Wort rechts vom Cursor

Durch "mit" wird in der Übersicht angedeutet, daß die **Pfeiltasten** immer gedrückt werden müssen, während die vorhergenannten Tasten niedergedrückt sind.

GEBRAUCH DER MICRODRIVES

Ein QL Programm wird normalerweise von Microdrive 1 geladen. Bevor Sie **Hilfe** anfordern oder auf einen Drucker ausgeben wollen, müssen Sie sich vergewissern, daß Microdrive 1 auch eine Kassette mit dem Programm enthält. Ansonsten können Sie die Kassette jederzeit herausnehmen, wenn die Anzeigelampe nicht gerade leuchtet.

Eine Kassette in Microdrive 2 – und in weiteren Microdrives – können Sie benutzen, um Informationen zu speichern; z. B. **QL Quill**-Dokumente, **QL Archive**-Dateien usw.

DATEINAMEN

Informationen werden auf Microdrive-Kassetten in Dateien gespeichert. Da eine Kassette mehrere Dateien enthalten kann, muß jeder Datei ein Name gegeben werden. Sonst könnten die verschiedenen Dateien ja nicht voneinander unterschieden werden. Ein Dateiname kann für die **QL Programme** aus höchstens acht Zeichen – Buchstaben, Ziffern, keine Leerzeichen – bestehen. Es ist zweckmäßig, "sinnvolle" Namen zu verwenden, die den Inhalt der Datei kurz charakterisieren. Z. B. ist "Verkauf" ein sinnvollerer Name für eine entsprechende Datei als "Hans".

Beim Laden und Sichern von Dateien nehmen die Programme standardmäßig an, daß sich die Kassette im Microdrive 2 befindet. Wollen Sie ein anderes Laufwerk, können Sie die gewünschte **Einheit** angeben. Am einfachsten sichern Sie eine Datei, indem Sie nach dem entsprechenden Kommando den Dateinamen eingeben. Z. B.:

```
Verkauf
```

Dadurch wird die Datei "Verkauf" standardmäßig auf Microdrive 2 gesichert. Soll sie dagegen auf Microdrive 1 gesichert werden, geben Sie ein:

```
mdv1_Verkauf
```

Ein weiterer Zusatz zum Dateinamen wird von den Programmen normalerweise automatisch eingefügt. Sie brauchen sich darum nicht zu kümmern. Dieser Zusatz kennzeichnet, welches Programm die Datei gesichert hat und um welche Art von Datei es sich handelt. Die folgenden Zusatzbezeichnungen können vorkommen und haben die angegebene Bedeutung:

_doc	ein Dokument, das von QL Quill erstellt wurde
_aba	ein Arbeitsblatt von QL Abacus
_grf	eine Datei, die von QL Easel erstellt wurde
_dbf	eine Datei von QL Archive
_prg	oder
_pro	ein Datenverwaltungsprogramm von QL Archive
_scn	eine Bildschirmmaske von QL Archive
_exp	eine Datei, die von einem QL Programm zur Weiterverarbeitung durch ein anderes Programm erstellt wurde
_lis	eine Datei, die von einem QL Programm zur späteren Ausgabe an einen Drucker erstellt wurde

Wenn Sie Informationen von einem QL Programm zu einem anderen übertragen wollen, dann wird eine besondere Datei erstellt. Ihr Zusatz lautet _exp (für Export). Alle Programme können Dateien mit diesem Zusatz verarbeiten. Der Informationsaustausch zwischen den einzelnen QL Programmen ist im Abschnitt *Information* unter *QL Programm – Import und Export* ausführlich beschrieben.

Sie können Ausgaben, die für einen Drucker vorgesehen sind, stattdessen auch in einer Datei zwischenspeichern, um sie erst später auszudrucken. Die QL Programme geben einer solchen Datei den Zusatz `_lis`.

VERZEICHNIS VON DATEIEN

Manche Kommandos der vier QL Programme erfordern die Angabe eines Dateinamens. Außer in Archive können Sie bei solchen Kommandos immer anfordern, daß ein Verzeichnis aller Dateien angezeigt wird. Das kann sehr sinnvoll sein, wenn Sie den genauen Namen einer Datei, die Sie gerade bearbeiten wollen, nicht mehr wissen.

Immer wenn ein Programm (außer Archive) die Eingabe eines Dateinamens erwartet, haben Sie die folgenden Möglichkeiten zur Auswahl:

- **ENTER** zu drücken, wenn Sie den Dateinamen wählen, den das Programm vorschlägt.
- Den Dateinamen, gefolgt von **ENTER** einzugeben.
- Ein **?** gefolgt von **ENTER** einzugeben, wenn Sie sich erst ein Verzeichnis aller Dateien ausgeben lassen wollen.

Wenn Sie ein Fragezeichen (gefolgt von **ENTER**) anstelle eines Dateinamens eingeben, fragt das Programm weiter:

```
Verz mdv2_
```

und schlägt damit vor, ein Verzeichnis aller Dateien von Microdrive 2 auszugeben. Sie können durch Eingabe von **ENTER** diesen Vorschlag akzeptieren oder Sie können den Namen der Einheit verändern (z. B. in `mdv1_`) und **ENTER** drücken, um sich das Verzeichnis einer anderen Einheit ausgeben zu lassen. Wenn das Verzeichnis ausgegeben ist, fordert das Programm von Ihnen die Eingabe des Dateinamens erneut an. Statt eines Dateinamens können Sie jetzt wieder ein ? eingeben, um sich das Verzeichnis einer anderen Kassette oder einer anderen Einheit anzeigen zu lassen.

Archive benutzt eine andere Methode. In Archive können Sie sich jederzeit mit dem Kommando **Verz** ein Verzeichnis von Dateien ausgeben lassen. Bei diesem Kommando müssen Sie dann noch die Einheit (z. B. "mdv1_") angeben, deren Dateiverzeichnis ausgegeben werden soll, oder gleich **ENTER** drücken, wenn Sie das Verzeichnis von **mdv2_** sehen wollen.

ESCAPE

In den vier QL Programmen können Sie jederzeit durch Drücken der **ESCape** Taste den zuletzt eingegebenen Befehl außer Kraft setzen. Anschließend können Sie die Arbeit an der Stelle des Programms fortsetzen, an der Sie das letzte Kommando eingegeben haben. Sie können **ESC** ebenso benutzen, um den augenblicklichen Inhalt der Eingabezeile vollständig zu löschen, oder um einen noch nicht vollständig ausgeführten Befehl abzubrechen.

ANDERE EINHEITEN

Alle Zusatzgeräte, auf die der QL Daten ausgeben oder von denen er Daten empfangen kann, nennen wir Einheiten. Außer von Microdrives können Daten auch von anderen Einheiten geladen oder auf anderen Einheiten gesichert werden. Die Einheit muß mit einem normalen SuperBASIC Namen bezeichnet werden. Näheres finden Sie unter Einheiten im Abschnitt Begriffe. Dem Namen muß ein Unterstreichungszeichen (_) vorangestellt werden.

Eine andere Einheit kann auch das QL-Netzwerk sein. Bevor Sie ein QL Programm laden, muß dann jedem Computer im Netzwerk eine Stationsnummer zugewiesen werden. Schalten Sie den Computer ein und legen Sie keine Kassette in Microdrive 1. Drücken Sie die Taste **F1** oder **F2**.

Um Ihrem Computer seine Stationsnummer zuzuteilen, geben Sie den Befehl **NET** ein, gefolgt von der Stationsnummer. Mit der folgenden Eingabe wird z. B. die Stationsnummer 5 zugeteilt.

```
NET 5 ENTER
```

Legen Sie dann die Programm-Kassette in Microdrive 1 ein. Das Programm starten Sie mit der Eingabe

```
lrun mdv1_boot ENTER
```

Wir geben zwei Beispiele für die Arbeit der **QL Programme** im **Netzwerk.** Im Netz befindet sich außer Station 5 noch eine Station mit der Nummer 1.

DRUCKEN ÜBER DAS NETZWERK

Station 5 hat mit **QL Quill** ein Dokument erstellt, das gedruckt werden soll. Ein Drukker befindet sich aber nur bei Station 1.

Station 5 gibt dann im Befehl Ausdruck anstatt Drucker ein:

```
_neto_1
```

Die Eingabezeile bei **QL Quill** sieht dann aus wie folgt:

```
Ausdruck,aktuell,ganz auf _neto_1,überschreiben
ja
```

Bevor der Ausdruck mit **ENTER** veranlaßt wird, muß Station 1 empfangsbereit sein. Dazu gibt sie in SuperBASIC folgenden Befehl ein:

```
copy neti_5 to ser1
```

Unmittelbar nachdem Station 5 **ENTER** eingibt, beginnt der Drucker seine Arbeit.

DATENAUSTAUSCH ÜBER DAS NETZWERK

Station 5 hat z. B. eine Kalkulationstabelle mit **QL Abacus** erstellt. Die Tabelle soll in ein Dokument eingefügt werden, das gerade bei Station 1 erstellt wird.

Im Dateienmenü von QL Abacus wird der **Export-Befehl** gewählt und als Einheitenname eingegeben:

```
_neto_1
```

Die Eingabezeile bei Abacus sieht dann aus wie folgt:

```
Dateien>Export zu quill,Bereich A1:D4,auf_neto_1,
überschreiben ja
```

Bevor **ENTER** eingegeben wird, muß Station 1 empfangsbereit sein. Station 1 wählt in QL Quill den Befehl Dateien, Import und gibt ein:

```
_neti_5
```

Nach **ENTER** erscheint die Frage "per Zeile", die mit **ENTER** bestätigt wird. Anschließend kann Station 5 mit **ENTER** den Exportvorgang einleiten. Nach kurzer Zeit steht die Kalkulationstabelle in dem aktuellen **QL Quill**-Dokument von Station 1.

IHR NEUER QL 1
DER GEBRAUCH DES HANDBUCHES . . 2
EINFÜHRUNG IN DEN QL 1
ANSCHLÜSSE 2
ANSCHLIESSEN 3
Das Netzteil 3
Der Bildschirm 3
Der QL Monitor 3
Das Fernsehgerät 3
ARBEITEN 4
DIE TASTATUR 4
Enter 4
Shift 4
Umschaltung 5
Löschen 5
Funktionstasten 5
DER BILDSCHIRM 5
MICRODRIVES 5
WEITERE ZUSATZGERÄTE 6
QL Printer 6
Diskettenlaufwerke 6
QL PRINTER 6
DIE NÄCHSTEN SCHRITTE 7
WENN PROBLEME AUFTRETEN 7
EINFÜHRUNG IN DIE QL PROGRAMME . . . 8
ARBEITSKOPIEN 8
LADEN 8
Andere Drucker statt QL Printer 9
BILDSCHIRMDARSTELLUNG 9
FUNKTIONSTASTEN 9
HILFE 10
DER ZEILENEDITOR 10
GEBRAUCH DER MICRODRIVES 11
DATEINAMEN 11
VERZEICHNIS VON DATEIEN 12
ESCAPE 12
ANDERE EINHEITEN 12
DRUCKEN ÜBER DAS NETZWERK 13
DATENAUSTAUSCH ÜBER DAS NETZWERK 13

sinclair

QL

Anfängerkurs

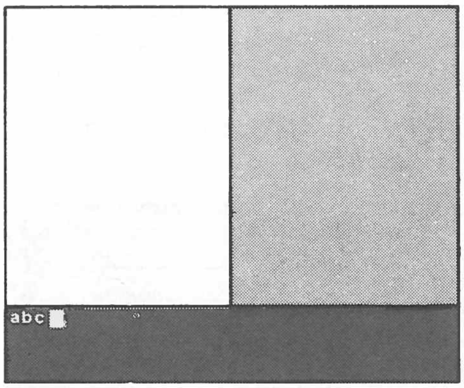
by Roy Atherton *(Bulmershe College Computer Centre)*

KAPITEL 1

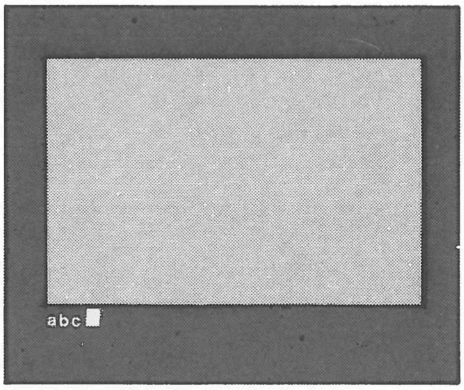
DIE ERSTEN SCHRITTE MIT DEM COMPUTER

BILDSCHIRM

Der QL muß an einen Monitor oder ein Fernsehgerät angeschlossen und eingeschaltet sein. Betätigen Sie nun einige Tasten, z. B. abc. Danach muß die Bildschirmanzeige wie nachfolgend dargestellt aussehen. Das kleine blinkende Licht wird als **Cursor** bezeichnet.



Monitor



Fernsehgerät

Sieht die Bildschirmanzeige nicht so aus, so lesen Sie bitte das Kapitel "Einführung". Dort steht, wie Sie gegebenenfalls Abhilfe schaffen können.

TASTATUR

Der QL ist ein vielseitiger und leistungsfähiger Computer, der auch Tastatur-Funktionen umfaßt, die Sie zu diesem Zeitpunkt noch nicht benötigen. Wir werden hier nur die Funktionen beschreiben, die Sie für dieses und die folgenden sechs Kapitel benötigen.

UNTERBRECHUNG

Damit können unerwünschte Situationen beendet werden. Zum Beispiel:

- eine Eingabe, die abgebrochen werden soll;
- es läuft etwas falsch, das Sie nicht verstehen;
- ein laufendes Programm, das vorzeitig beendet werden soll;
- irgendein anderes Problem.

Da eine Unterbrechung sehr folgenschwer sein kann, wurde verhindert, daß sie versehentlich ausgelöst wird.

Halten Sie die [CTRL]-Taste gedrückt und betätigen Sie dann die [LEERTASTE]

Wurden zu einem Programm, das mit der **BREAK**-Funktion angehalten wurde, keine Anweisungen hinzugefügt oder keine Anweisungen gelöscht, so kann das Programm durch Eingabe von:

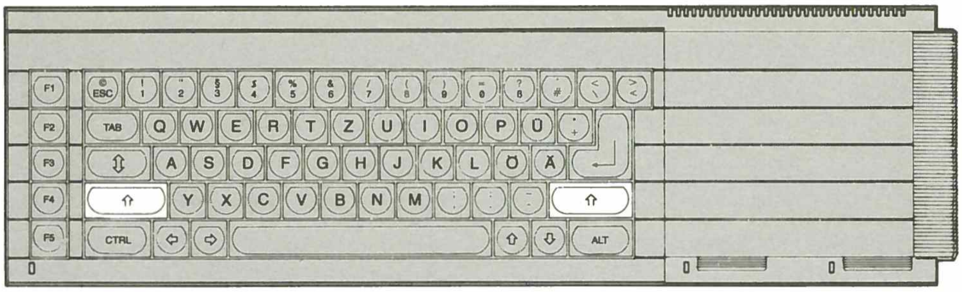
```
CONTINUE
```

erneut gestartet werden.

RÜCKSETZUNG

Hier handelt es sich nicht um eine Taste auf der Tastatur, sondern um eine kleine Drucktaste auf der rechten Seite des QL. Sie wurde absichtlich schwer zugänglich angebracht, da ihre Auswirkungen wesentlich weitreichender sind, als die Unterbrechung. Kann das gewünschte Ergebnis nicht mit den Tasten zur Unterbrechung erzielt werden, so muß die **RÜCKSETZ**-Taste betätigt wirden. Dies ist nahezu gleichbedeutend mit dem Ausschalten und Wiedereinschalten des Computers. Durch Betätigung dieser Taste wird ein Neustart vorgenommen.

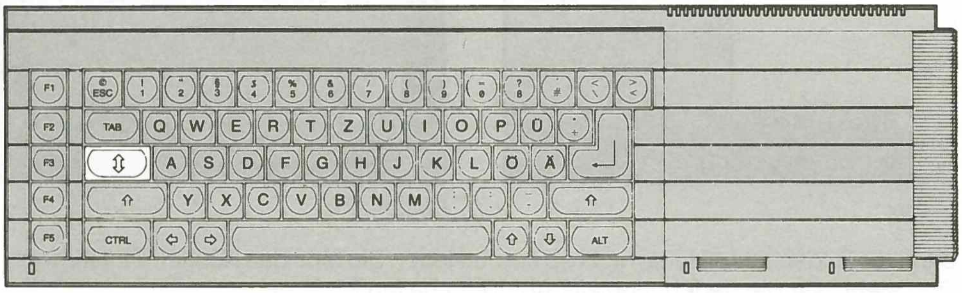
SHIFT-TASTEN



Es gibt zwei **SHIFT**-Tasten, da sie häufig benutzt werden und sowohl mit der rechten, als auch mit der linken Hand zugänglich sein müssen.

Eine **SHIFT**-Taste wird gedrückt gehalten, während einige Buchstabentasten betätigt werden. Auf diese Weise werden Großbuchstaben ausgegeben.
Eine **SHIFT**-Taste wird gedrückt gehalten und eine andere Taste als eine Buchstabentaste wird betätigt. Auf diese Weise wird das im oberen Teil der Tasten eingravierte Symbol erhalten.
Ohne Betätigung einer **SHIFT**-Taste erhalten Sie Kleinbuchstaben oder das Symbol, das im unteren Teil der Taste eingraviert ist.

FESTSTELL-TASTE



Diese Taste wird wie ein Schalter benutzt. Wird sie ein Mal betätigt, so werden nur die Buchstabentasten in einem bestimmten Modus "blockiert" – im Groß- oder Kleinschreibe-Modus.

Betätigen Sie einige Buchstabentasten.
Betätigen Sie die **FESTSTELL**-Taste ein Mal.
Betätigen Sie einige Buchstabentasten.

Hier werden Sie feststellen, wie sich der Modus ändert und wirksam bleibt, bis die **FESTSTELL**-Taste erneut betätigt wird.

LEERTASTE



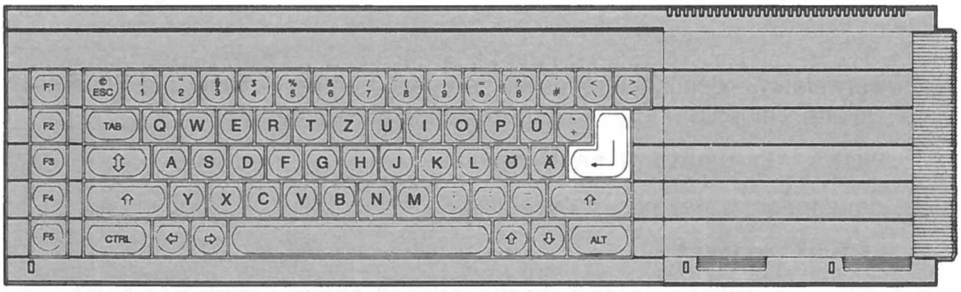
Die lange Taste in der untersten Tastenreihe ist die Leertaste. Wie Sie aus Kapitel 2 ersehen werden, ist dies eine sehr wichtige Taste bei SuperBAISC.

LÖSCHEN



Die Pfeil-Taste nach links zusammen mit der **CTRL**-Taste kann als Löschtaste benutzt werden. Die **CTRL**-Taste muß bedrückt gehalten werden, während die Cursor-Taste betätigt wird. Werden diese beiden Tasten gemeinsam betätigt, so wird das vorhergehende Zeichen gelöscht.

ENTER



Das System muß wissen, wann Sie eine Daten- oder Befehlseingabe abgeschlossen haben. Haben Sie beispielsweise den **RUN**-Befehl vollständig eingegeben, so betätigen Sie die **ENTER**-Taste, um seine Ausführung zu veranlassen.

Da diese Taste so häufig benutzt wird, haben wir ihr ein besonderes Symbol zugewiesen:

←

Dieses Symbol wird der Bequemlichkeit halber, zur deutlicheren Darstellung und um Platz zu sparen benutzt. Testen Sie nun die Funktion der **ENTER**-Taste, indem Sie:

```
PRINT "Richtig" ←
```

eingeben. Kommt es zu keinem Fehler, so antwortet das System mit:

```
Richtig
```

ANDERE TASTATUR-SYMBOLE ZUR DIREKTEN BENUTZUNG

*	Multiplikation	+	Addition
_	Unterstreichungs-zeichen	=	Zuweisungszeichen (wird in LET-Anweisungen benutzt)
"	Anführungszeichen	'	Apostroph
,	Komma	!	Ausrufungszeichen
;	Semikolon	&	Verkettung (Strings)
:	Doppelpunkt	.	Dezimalpunkt oder Punkt
\	Umgekehrter Schrägstrich	$	Dollarzeichen
(	Klammer auf	)	Klammer zu

GROSS- UND KLEIN-SCHREIBUNG

SuperBASIC erkennt Befehle unabhängig davon, ob sie mit Groß- oder Kleinbuchstaben geschrieben werden. So kann beispielsweise der SuperBASIC-Befehl für das Löschen des Bildschirms **CLS** folgendermaßen eingegeben werden:

```
cLS ←
cls ←
CLS ←
```

All diese Schreibweisen sind korrekt und haben dieselbe Auswirkung. Einige Befehle werden zum Teil mit Großbuchstaben angezeigt, um zulässige Abkürzungen anzugeben. Kann ein Befehl nicht abgekürzt werden, so wird er vollständig in Großbuchstaben angezeigt.

BENUTZUNG VON ANFÜHRUNGSZEICHEN

Normalerweise werden Anführungszeichen dazu benutzt, Anfang und Ende einer Zeichenkette – eines Strings – zu kennzeichnen. Versuchen Sie:

```
PRINT "Es funktioniert" ⇐
```

Der Computer antwortet mit:

```
Es funktioniert
```

Die Anführungszeichen werden nicht ausgedruckt, sondern markieren Anfang und Ende des Textes, der ausgedruckt werden soll. Soll das Anführungszeichen selbst ausgedruckt werden, so können als Markierung von Anfang und Ende des zu druckenden Textes Apostrophe verwendet werden. Zum Beispiel:

```
PRINT 'Das Anführungszeichen ist" ' ⇐
```

Hier wird folgendes ausgedruckt:

```
Das Anführungszeichen ist "
```

HÄUFIGE TIPPFEHLER

Die Null-Taste befindet sich bei den anderen Zifferntasten in der obersten Reihe der Tastatur.

Die Taste für den Buchstaben "O" befindet sich bei den anderen Buchstabentasten. Das "O" ist etwas dicker als die Null. Hier muß darauf geachtet werden, daß das richtige Symbol benutzt wird.

Gleichermaßen muß eine Verwechslung zwischen Eins für die Ziffern und dem Buchstaben "I" bei den Buchstabentasten vermieden werden.

BETÄTIGUNG DER SHIFT-TASTE

Wird eine **SHIFT**-Taste benutzt, so muß sie gedrückt gehalten werden, während die andere Taste betätigt wird. Das heißt, daß die **SHIFT**-Taste vor der anderen Taste betätigt wird und erst nach der anderen Taste losgelassen wird.

Dieselbe Regel gilt für die **CTRL**- und **ALT**-Tasten, die zusammen mit anderen Tasten benutzt werden. Sie werden hier jedoch noch nicht benötigt.

Geben Sie die beiden folgenden einfachen Befehle ein:

```
CLS ⇐
PRINT 'Guten Tag' ⇐
```

Genau genommen handelt es sich bei diesen Befehlen um ein Computerprogramm. Bei der Arbeit mit Computern ist jedoch das **gespeicherte Programm** von großer Bedeutung. Die obigen Befehle werden sobald Sie die Eingabe mit ⇐ (ENTER) abschließen, ausgeführt.

Nun geben Sie das Programm mit Zeilennummern ein:

```
10 CLS ⇐
20 PRINT 'Guten Tag'
```

Diesmal geschieht zuerst nichts, sieht man davon ab, daß das Programm im oberen Bildschirmteil angezeigt wird. Dies bedeutet, daß es keine erkennbaren syntaktischen Fehler enthält. Es entspricht den SuperBASIC-Regeln, wurde jedoch noch nicht ausgeführt, sondern nur gespeichert. Um dieses Programm auszuführen, geben Sie:

```
RUN ⇐
```

ein. Der Unterschied zwischen direkten Befehlen zur sofortigen Ausführung und einer gespeicherten Befehlsfolge wird im nächsten Kapitel erläutert. Für den Augenblick sollten Sie mit den obigen Beispielen und zwei weiteren Beispielen etwas experimentieren:

LIST ⇐

Mit diesem Befehl wird ein intern gespeichertes Programm auf dem Bildschirm oder auf einer anderen Einheit angezeigt (aufgelistet).

NEW ⇐

Mit diesem Befehl wird ein intern gespeichertes Programm gelöscht, so daß Sie ein **NEUES** Programm eingeben können.

TEST ZU KAPITEL 1

Sie können maximal 16 Punkte erzielen. Prüfen Sie Ihr Ergebnis anhand der Antworten auf Seite 107.

1. Unter welchen Umständen benutzen Sie die Tastenkombination für die Unterbrechung?
2. Wo befindet sich die RÜCKSETZ-Taste?
3. Wie wirkt sich die RÜCKSETZ-Taste aus?
4. Geben Sie zwei Unterschiede zwischen einer **SHIFT**-Taste und der **FESTSTELL**-Taste an.
5. Wie können Sie ein gerade eingegebenes falsches Zeichen löschen?
6. Welche Funktion hat die **ENTER**-Taste?
7. Welches Symbol wird für die **ENTER**-Taste benutzt?

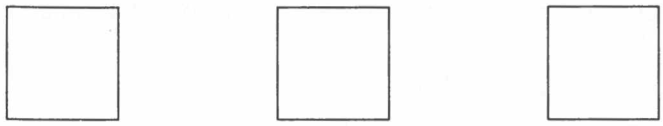
Welche Funktionen haben die Befehle in den Fragen 8 bis 11?

8. **CLS** ⇐
9. **RUN** ⇐
10. **LIST** ⇐
11. **NEW** ⇐
12. Haben die Befehle auch in Kleinbuchstaben noch die richtige Bedeutung?
13. Welche Bedeutung haben die Teile der Befehle, die von dem QL in Großbuchstaben aufgelistet werden?

KAPITEL 2

AN-WEISUNGEN AN DEN COMPUTER

Bei Computern müssen oft Daten in Form von Zahlen gespeichert werden. Die Speicherung kann mit Ablagefächern verglichen werden.



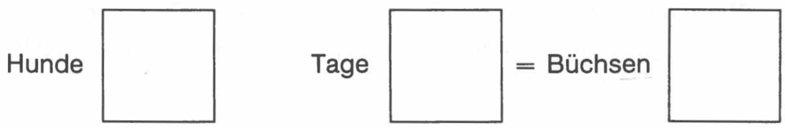
Auch wenn Sie die einzelnen Ablagefächer nicht sehen können, müssen Sie ihnen doch Namen zuweisen. Angenommen, Sie möchten hier die folgende einfache Berechnung anstellen.

Ein Hundezüchter muß neun Hunde während 28 Tagen füttern, wobei jeder pro Tag eine Büchse "Happi" erhält. Der Computer soll nun die benötigte Anzahl von Büchsen ausdrucken (auf dem Bildschirm anzeigen).

Bei einer Möglichkeit zur Lösung dieses Problems sind drei Ablagefächer für:

- Anzahl von *Hunden*
- Anzahl von *Tagen*
- Gesamtzahl von *Büchsen*

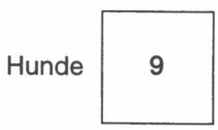
erforderlich. Bei SuperBASIC können Sie bedeutungsvolle Namen für die Ablagefächer wählen, z. B.:



Der Computer kann nun mit einem einzigen Befehl angewiesen werden, ein Ablagefach zu erstellen, dieses zu benennen und eine Zahl in dem Ablagefach zu speichern:

```
LET hunde = 9 ←
```

Mit diesem Befehl wird ein internes Ablagefach namens *Hunde* erstellt. In dieses Ablagefach wird die Zahl 9 gesetzt:



Manche Worte, wie zum Beispiel **LET** haben bei SuperBASIC eine besondere Bedeutung. Es wird als Schlüsselwort bzw. **Befehl** bezeichnet. SuperBASIC verfügt noch über viele andere Befehle, wie Sie noch feststellen werden. Es muß jedoch stets auf das Leerzeichen hinter **LET** und den anderen Schlüsselwörtern geachtet werden. Da Sie bei SuperBASIC Namen für Ablagefächer relativ frei auswählen können, wäre *LEThund* ein gültiger Name für ein Ablagefach.

In SuperBASIC ist es nicht erforderlich, das Schlüsselwort **LET** einzugeben. Deshalb sind Anweisungen wie:

```
LEThunde = 9 ←
```

zulässig. Dadurch wird auf ein Ablagefach namens **LEThunde** Bezug genommen.

Auf diese Weise müssen, wie bei anderen Sprachen, Namen, Zahlen und Befehle durch Leerzeichen voneinander getrennt werden, sofern sie nicht durch Sonderzeichen voneinander getrennt werden.

Selbst wenn dies nicht erforderlich wäre, würde eine Programmzeile ohne die richtigen Leerzeichen schlecht aussehen. Bei Computern mit einer kleinen Speicherkapazität mögen Programmierer hierzu gezwungen sein, bei dem QL ist dies jedoch nicht der Fall.

Sie können prüfen, ob ein Ablagefach eingerichtet wurde, indem Sie:

```
PRINT hunde ←
```

eingeben. In diesem Fall muß der Inhalt des Ablagefaches auf dem Bildschirm angezeigt werden:

```
9
```

Auch hier beachten Sie bitte wieder das Leerzeichen hinter **PRINT**.

Zur Lösung des Problems können wir ein Programm schreiben, das aus einer Folge von Anweisungen besteht. Die beiden ersten Befehle kennen Sie schon:

```
LET hunde = 9 ←
LET tage = 28 ←
```

Mit diesen beiden Befehlen werden Ablagefächer erstellt, benannt und werden ihnen Zahlen oder Werte zugewiesen.

Mit dem nächsten Befehl muß eine Multiplikation ausgeführt werden. Hierzu benutzt der Computer das Symbol *. Das Ergebnis der Multiplikation muß in ein neues Ablagefach namens *Büchsen* gestellt werden:

```
LET büchsen = hunde * tage ←
```

1. Der Computer ermittelt die Werte, 9 und 28, aus den beiden Ablagefächern namens *Hunde* und *Tage*.
2. Die Zahl 9 wird mit 28 multipliziert.
3. Ein neues Ablagefach wird erstellt und erhält den Namen *Büchsen*.
4. Das Ergebnis der Multiplikation wird als Wert in das Ablagefach namens *Büchsen* gestellt.

All dies scheint klar. Allerdings müssen Sie unbedingt die zugrundeliegenden Ideen verstehen.

Nun braucht der Computer nur noch das Ergebnis auszudrucken. Hierzu wird:

```
PRINT büchsen ←
```

eingegeben. Dadurch wird:

```
252
```

auf dem Bildschirm angezeigt.

Zusammenfassend erzeugt das Programm:

```
LET hunde = 9 ←
LET tage = 28 ←
LET büchsen = hunde * tage ←
LET büchsen ←
```

folgende interne Effekte, die man sich am besten als drei Ablagefächer mit Zahlen vorstellt:

Hunde [9] × Tage [28] = Büchsen [252]

Außerdem kommt es zu folgender Ausgabe auf dem Bildschirm:

```
252
```

Natürlich könnte dieses Ergebnis problemloser mit einem Taschenrechner oder einem Bleistift und einem Blatt Papier erzielt werden. Sie könnten dies schnell mit dem QL erreichen, indem Sie:

```
PRINT 9 * 28 ←
```

eingeben. Dadurch würde die Antwort auf dem Bildschirm angezeigt. Die hier dargelegten Ideen stellen jedoch wichtige Grundlagen bei der Programmierung in SuperBASIC dar. Sie sind so grundlegend, daß sie in vielen Computersprachen auftreten und über eigene Namen verfügen.

1. Bezeichnung wie *Hunde, Tage* und *Büchsen* werden als **Namen** bezeichnet.
2. Ein einzelner Befehl wie beispielsweise:

   ```
   LET hunde = 9 ⏎
   ```

   wird als **Anweisung** bezeichnet.
3. Ein Ablagefach, dem ein Name zugeordnet ist, wird als **Variable** bezeichnet. Durch Ausführung der obigen Anweisung wird der Wert 9 in das Ablagefach gespeichert, das den Namen *Hunde* trägt.

Eine Anweisung wie:

```
LET hunde = 9 ⏎
```

ist ein Befehl für ein überaus dynamisches internes Verfahren. Der ausgedruckte Text ist jedoch statisch und benutzt das aus der Mathematik stammende = Zeichen. Man könnte auch:

```
LET Hunde wird zu 9
```

denken (jedoch nicht eingeben). Für das Verfahren selbst stellt man sich eine Richtung von rechts nach links vor (bitte nicht so eingeben):

```
hunde ← 9
```

Das Zeichen "=" in einer **LET**-Anweisung ist nicht gleichbedeutend mit dem in der Mathematik üblichen Gleichheitszeichen. Nimmt unser Hundezüchter beispielsweise noch einen weiteren Hund auf, so könnte man folgendes schreiben:

```
LET hunde = hunde + 1 ⏎
```

Dies ist keine Gleichsetzung im mathematischen Sinn. Sie wäre ja nicht besonders sinnvoll. Für den Computer ist dies jedoch eine einfache Anweisung. War der Wert für Hunde vor der Zuweisung gleich 9, so wird der Wert durch die Zuweisung auf 10 gesetzt. Testen Sie dies, indem Sie folgendes Programm eingeben:

```
LET hunde = 9 ⏎
PRINT hunde ⏎
LET hunde = hunde + 1 ⏎
PRINT hunde ⏎
```

Nun muß:

```
9
10
```

ausgegeben werden. Damit wird bewiesen, daß der endgültige Wert in dem Ablagefach gleich:

Hunde	**10**

ist. Um zu verstehen, was mit den Ablagefächern oder Variablen geschieht, wird am besten ein sogenannter "Trockenlauf" ausgeführt. Sie überprüfen einfach die Befehle nacheinander und schreiben die sich aus jedem Befehl ergebenden Werte auf, um zu sehen, wie die Ablagefächer erstellt wurden, wie ihnen Werte zugewiesen wurden und wie sie ihre Werte während der Programmausführung verändern.

```
LET hunde = 9 ⏎
LET tage = 28 ⏎
LET büchsen = hunde * tage ⏎
PRINT büchsen ⏎
```

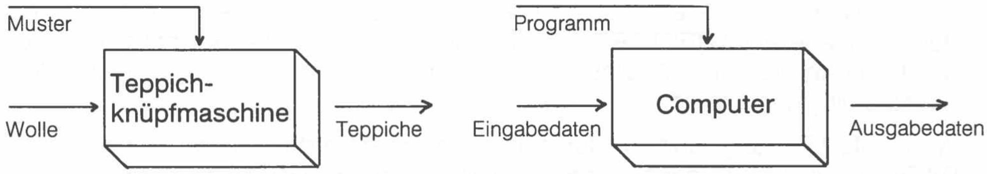
Hier muß:

```
252
```

ausgegeben werden. Sie werden schon festgestellt haben, daß ein Variablennamen bis jetzt immer zuerst auf der linken Seite einer **LET**-Anweisung benutzt wurde. Sobald das Ablagefach erstellt und ihm ein Wert zugewiesen wurde, kann der entsprechende Variablenname auch auf der rechten Seite einer **LET**-Anweisung benutzt werden.

EINGABE – INPUT, READ UND DATA

Eine Teppichknüpfmaschine benötigt Wolle als Eingabe. Sie stellt dann Teppiche in den ausgewählten Mustern her.



Wird die Wolle ausgewechselt, so erhält man unter Umständen einen anderen Teppich.

Ähnliche Beziehungen bestehen auch in einem Computer.

Werden jedoch die Daten mit Hilfe von **LET** in Ablagefächer eingegeben, so gibt es zwei Nachteile, sobald die Grenze der einfachen Programme überschritten ist:

- das Schreiben von **LET**-Anweisungen ist arbeitsaufwendig,
- die Änderung dieser Eingabe ist ebenfalls arbeitsaufwendig.

Daten können auch in ein Programm eingegeben werden, während dieses ausgeführt wird. Mit der **INPUT**-Anweisung pausiert das Programm und wartet, bis Sie eine Eingabe über die Tastatur vornehmen.

Als erstes geben Sie:

```
NEW ↵
```

ein, so daß das vorher gespeicherte Programm (sofern es noch vorhanden ist) gelöscht wird und der Speicher für das neue Programm vorbereitet ist. Nun geben Sie:

```
100 LET preis = 15 ↵
110 PRINT "Wieviele Stifte?" ↵
120 INPUT stifte ↵
130 LET kosten = preis * stifte ↵
140 PRINT kosten ↵
RUN ↵
```

ein. Das Programm pausiert bei Zeile 120. An dieser Stelle müssen Sie die gewünschte Anzahl von Stiften eingeben, z. B.:

```
4 ↵
```

Vergessen Sie die **ENTER**-Taste nicht. Nun wird:

```
60
```

ausgegeben. Mit der **INPUT**-Anweisung muß ein Variablenname angegeben werden, so daß das System weiß, wohin es die über die Tastatur eingegebenen Daten stellen soll. Zeile 120 zusammen mit der von Ihnen vorgenommenen Eingabe hat dieselbe Wirkung wie eine **LET**-Anweisung. In den Fällen, in denen eine Zusammenarbeit zwischen dem Computer und dem Benutzer gewünscht wird, ist sie bequemer. Die **LET**- und **INPUT**-Anweisungen sind jedoch nur bei kleinen Datenmengen nützlich. Für größere Datenmengen werden andere Anweisungen ohne Pausen bei der Ausführung des Programms benötigt.

Bei SuperBASIC, wie bei den anderen BASIC-Dialekten, gibt es eine andere Eingabemethode, bei der mit **READ** aus **DATA**-Anweisungen gelesen wird. Das obigen Programm kann in einer neuen Form eingegeben werden und erzielt dieselben Ergebnisse ohne Pause. Versuchen Sie dies:

```
NEW ↵
100 READ preis, stifte ↵
110 LET kosten = preis * stifte ↵
120 PRINT kosten ↵
130 DATA 15,4 ↵
RUN ↵
```

Nun muß wie vorher:

```
60
```

ausgegeben werden.

Bei jeder Ausführung des Programms muß SuperBASIC mitgeteilt werden, an welcher Stelle mit dem Lesen der Daten begonnen werden soll. Dies erfolgt entweder durch Eingabe eines **RESTORE**-Befehls gefolgt von der **DATA**-Zeilennummer oder durch Eingabe von **CLEAR**. Diese beiden Befehle können auch am Anfang des Programms eingefügt werden.

Wird Zeile 100 ausgeführt, so durchsucht das System das Programm nach einer **DATA**-Anweisung. Danach werden die Werte in der **DATA**-Anweisung für die Variablen in der **READ**-Anweisung in genau derselben Reihenfolge benutzt. **DATA**-Anweisungen werden vom Programm benutzt, werden jedoch nicht in dem Sinne ausgeführt, wie die einzelnen anderen Zeilen nacheinander ausgeführt werden. **DATA**-Anweisungen können an beliebiger Stelle in einem Programm stehen, werden jedoch am besten an das Ende eines Programms gestellt. Sie sind für das aktive Programm wichtig, jedoch nicht Bestandteil dieses aktiven Programms. Für **READ** und **DATA** gelten folgende Regeln:

1. Sämtliche **DATA**-Anweisungen werden als eine einzige lange Folge von Elementen betrachtet. Bis jetzt handelte es sich bei diesen Elementen um Zahlen, sie können jedoch auch Wörter oder Buchstaben darstellen.
2. Bei jeder Ausführung einer **READ**-Anweisung werden die Werte aus der **DATA**-Anweisung in die Variablen kopiert, die in der **READ**-Anweisung angegeben werden.
3. In einem internen Eintrag verfolgt das System, welche Elemente mit Hilfe von **READ** gelesen wurden. Versucht ein Programm mehr Elemente zu lesen, als in allen **DATA**-Anweisungen vorhanden sind, so wird ein Fehler angezeigt.

NAMEN

Für die "Ablagefächer" wurden Namen wie *Hunde, Tage* usw. benutzt. Für diese Namen gibt es jedoch bestimmte Regeln:

- Ein Name darf keine Leerzeichen enthalten.
- Ein Name muß mit einem Buchstaben beginnen.
- Ein Name muß aus **Buchstaben, Ziffern, $, %, _** (Unterstreichungszeichen) bestehen.
- Die Symbole **$, %** haben eine besondere Bedeutung. Sie wird später erläutert. Das **Unterstreichungszeichen** kann jedoch benutzt werden, um Namen wie:

  Hunde_futter
  Gesamt_monats_löhne

  leserlicher zu gestalten.
- SuperBASIC unterscheidet nicht zwischen Groß- und Kleinbuchstaben. Deshalb sind Namen wie *BÜCHSE* und *büchse* gleichbedeutend.
- Ein Name kann maximal 255 Zeichen umfassen.

Nach diesen Regeln erstellte Namen können in SuperBASIC auch noch für andere Zwecke benutzt werden. Diese Regeln lassen eine große Flexibilität bei der Auswahl der Namen zu, so daß die Programme verständlicher gestaltet werden können. Namen wie *Gesamt, Zähler, Stifte* sind bedeutungsvoller als Namen wie Z, P, Q.

Nun wollen wir ein Kind zum Sparen erziehen. Für jede gesparte Mark erhält das Kind zwei Riegel Schokolade. Dies soll nun wie folgt berechnet werden:

```
LET tafeln = DM * 2 ←
PRINT tafeln ←
```

Mit diesem Programm können Sie noch keinen Testlauf ausführen, da Sie nicht wissen, wie viele DM gespart wurden.

Hier haben wir mit Absicht einen Fehler aufgenommen, indem wir *Preis* auf der rechten Seite einer **LET**-Anweisung benutzt haben, ohne ein Ablagefach erstellt und einen Wert zugewiesen zu haben. Der QL sucht intern nach der Variablen *Preis*. Er findet sie nicht und folgert daraus, daß das Programm einen Fehler enthält. Eine Fehlermeldung wird ausgegeben. Hätten wir versucht, den Wert von *Preis* auszudrukken, so hätte der QL ein * ausgedruckt, um anzugeben, daß *Preis* nicht definiert ist. Hier sagt man, daß die Variable *Preis* nicht **initialisiert** ist (d. h. daß ihr kein Ausgangswert zugewiesen wurde). Das Programm wird einwandfrei ausgeführt, wenn Sie zuerst folgende Schritte ausführen:

Tafeln	DM
7	
7	14

```
LET DM = 7 ←
LET tafeln = DM * 2 ←
```

Das Programm wird einwandfrei ausgeführt und gibt:

```
14
```

aus.

EIN GESPEICHERTES PROGRAMM

Die Eingabe von Anweisungen ohne Zeilennummern kann durchaus zu dem gewünschten Ergebnis führen. Es gibt jedoch zwei Gründe, warum diese – bis jetzt benutzte – Methode nur zu Einführungszwecken zufriedenstellend ist.

1. Das Programm kann nur so schnell arbeiten, wie Sie die Befehle eingeben. Dies ist bei einem Computer, der Millionen von Operationen pro Sekunde ausführen kann, nicht sehr beeindruckend.
2. Die einzelnen Befehle werden nach der Ausführung nicht gespeichert, so daß Sie das Programm nicht noch einmal ausführen lassen oder einen Fehler korrigieren können, ohne das ganze Programm neu einzugeben.

Charles Babbage, ein Computer-Pionier des 19. Jahrhunderts, wußte, daß ein erfolgreicher Computer Befehle und Daten in internen Ablagefächern speichern muß. Diese Befehle können dann in rascher Folge ohne weiteren Eingriff des Bedieners ausgeführt werden.

Werden Zeilennummern benutzt, so werden die Programmbefehle gespeichert, jedoch nicht sofort ausgeführt. Versuchen Sie folgendes:

```
100 LET preis = 15 ←
110 LET stifte = 7 ←
120 LET kosten = preis * stifte ←
130 PRINT kosten ←
```

Extern geschieht hier nichts. Das ganze Programm wird jedoch im Computer gespeichert. Sie können es ausführen, indem Sie:

```
RUN ←
```

eingeben. Danach muß:

```
105
```

angezeigt werden.

Der Vorteil dieser Lösung besteht darin, daß Sie das Programm mit einem minimalen Eingabeaufwand bearbeiten oder erweitern können.

BEARBEITEN EINES PROGRAMMS

Die vollständigen Bearbeitungsfunktionen von SuperBASIC werden später im einzelnen beschrieben. Jedoch selbst in diesem frühen Stadium können Sie drei Dinge problemlos ausführen:

- eine Zeile ersetzen
- eine neue Zeile einfügen
- eine Zeile löschen.

Ersetzen einer Zeile

Angenommen, das vorhergehende Programm soll geändert werden, da der Preis für einen Stift auf 20 Pfennig erhöht wurde. Hierzu geben Sie einfach Zeile 10 neu ein.

```
100 Let preis = 20 ⟸
```

Diese Zeile ersetzt die alte Zeile 10. Angenommen, die anderen Zeilen sind noch gespeichert. Testen Sie das Programm durch folgende Eingabe:

```
RUN ⟸
```

Jetzt muß 140 auf dem Bildschirm angezeigt werden.

Einfügen einer neuen Zeile

Angenommen, Sie möchten direkt vor der letzten Zeile eine Zeile einfügen, mit der "Gesamtkosten" ausgedruckt werden soll. Diese Situation tritt sehr häufig ein, so daß normalerweise Zeilennummern von 100, 110, 120... gewählt werden, um Platz für zusätzliche Zeilen zu lassen.

Für die neue Zeile geben Sie:

```
125 PRINT ''Gesamtkosten'' ⟸
```

ein. Diese Zeile wird dann direkt vor Zeile 130 eingefügt. Das System läßt Zeilennummern im Bereich von 1 bis 32768 zu, so daß die Zeilennummern sehr flexibel ausgewählt werden können. Man kann sich doch schwer im voraus sicher sein, welche Änderungen zu irgendeinem Zeitpunkt erforderlich sind.

Nun geben Sie:

```
RUN ⟸
```

ein. Die neue Ausgabe muß folgendermaßen aussehen:

```
Gesamtkosten
140
```

Löschen einer Zeile

Sie können Zeile 125 löschen, indem Sie:

```
125
```

eingeben. Zeile 125 wird dadurch aus dem Programm entfernt.

AUSGABE – PRINT

Hier werden Sie sehen, wie nützlich die **PRINT**-Anweisung ist. Sie können einen Text mit **PRINT** ausdrucken, indem Sie ihn in Anführungszeichen oder Apostrophe setzen:

```
PRINT ''Schokoladentafeln'' ⟸
```

Die Werte von Variablen (der Inhalt der Ablagefächer) kann ausgedruckt werden, indem Anweisungen wie:

```
PRINT tafeln ⟸
```

ohne Anführungszeichen eingegeben werden.

Sie werden später noch sehen, wie vielseitig die **PRINT**-Anweisung in SuperBASIC sein kann. Mit ihr können Sie Text oder andere Ausgaben genau an der gewünschten Stelle auf den Bildschirm setzen. Für den Augenblick reichen uns jedoch die beiden folgenden Funktionen aus:

- Ausdrucken von Texten
- Ausdrucken der Werte von Variablen (d. h. des Inhalts von Ablagefächern).

TEST ZU KAPITEL 2

Bei diesen Tests können Sie maximal 20 Punkte erreichen. Prüfen Sie Ihre Punktzahl anhand der Antworten auf Seite 108.

1. Womit kann die interne Speicherung einer Zahl verglichen werden?
2. Geben Sie zwei Arten an, mit denen Werte in Ablagefächern gespeichert werden können (zwei Punkte).
3. Wie können Sie den Wert eines Ablagefaches ermitteln?
4. Wie lautet der Fachausdruck für "Ablagefach"?
5. Wann erhält ein Ablagefach seinen ersten Wert?
6. Eine Variable wird als Variable bezeichnet, da sich ihr Wert während der Ausführung des Programms ändern kann. Auf welchem Weg wird eine derartige Änderung normalerweise vorgenommen?
7. Das "="-Zeichen in einer **LET**-Anweisung bedeutet nicht "gleich" wie in der Mathematik. Was ist die Bedeutung des "="-Zeichens?
8. Was geschieht, wenn Sie die **ENTER**-Taste bei einer Anweisung ohne Zeilennummer betätigen?
9. Was geschieht, wenn Sie die **ENTER**-Taste bei einer Anweisung mit Zeilennummer betätigen?
10. Welche Bedeutung haben die Anführungszeichen in einer **PRINT**-Anweisung?
11. Was geschieht, wenn keine Anführungszeichen in einer **PRINT**-Anweisung benutzt werden?
12. Was geschieht in einer **INPUT**-Anweisung im Gegensatz zu einer **LET**-Anweisung?
13. Welche Art von Programmanweisung wird niemals ausgeführt?
14. Was ist die Aufgabe einer **DATA**-Anweisung?
15. Schreiben Sie drei gültige Namen mit Buchstaben, Buchstaben und Ziffern, Buchstaben und Unterstreichungszeichen auf (drei Punkte).
16. Warum ist die Leertaste bei SuperBASIC besonders wichtig?
17. Warum sind frei auswählbare Namen bei der Programmierung wichtig?

AUFGABEN ZU KAPITEL 2

1. Führen Sie einen Trockenlauf aus, um die Werte sämtlicher Variablen anzuzeigen, während das folgende Programm ausgeführt wird:

```
100 LET stunden = 40 ⟵
110 LET stundenlohn = 15 ⟵
120 LET gehalt = stunden * stundenlohn ⟵
130 PRINT stunden, stundenlohn, gehalt ⟵
```

2. Schreiben und testen Sie ein Programm, wie bei Aufgabe 1, mit dem die Fläche eines Teppichs mit einer Größe von 3×4 berechnet wird. Benutzen Sie die Variablennamen: *Breite, Länge, Fläche.*
3. Schreiben Sie das Programm von Aufgabe 1 so um, daß anstelle der **LET**-Anweisungen zwei **INPUT**-Anweisungen benutzt werden.
4. Schreiben Sie das Programm in Aufgabe 1 so um, daß die Eingabedaten (40 und 3) in einer **DATA**-Anweisung anstelle einer **LET**-Anweisung stehen.
5. Schreiben Sie das Programm in Aufgabe 2 mit einer anderen Methode der Dateneingabe neu. Benutzen Sie **READ** und **DATA**, wenn Sie ursprünglich **LET** benutzt haben, und umgekehrt.
6. Hans und Peter wollen ein Spiel machen. Jeder nimmt alle Markstücke aus seinem Geldbeutel und gibt sie dem Partner. Schreiben Sie ein Programm, in dem dieser Vorgang vollständig mit **LET**- und **PRINT**-Anweisungen simuliert wird. Eine dritte Person, Susanne, nimmt das Geld von Hans, während Hans Peters Geld nimmt.
7. Schreiben Sie das Programm in Aufgabe 6 so um, daß die beiden auszutauschenden Zahlen in einer **DATA**-Anweisung stehen.

KAPITEL 3

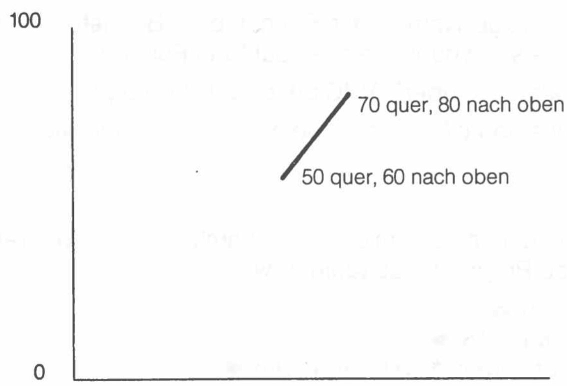
AUF DEM BILDSCHIRM ZEICHNEN

Für die Benutzung eines Fernsehgerätes oder Monitors mit dem QL gibt es zwei verschiedene Bildschirm-Betriebsarten. Zwischen ihnen wird mit dem **MODE**-Befehl ausgewählt. Mit **MODE 8** wählen Sie die niedrige Auflösung mit 256 × 256 Pixeln und der Möglichkeit, acht Vollfarben darzustellen. In dieser Betriebsart können in einer Zeile bis zu 42 Zeichen ausgegeben werden. Mit **MODE 4** wählen Sie die hochauflösende Betriebsart mit 512 × 256 Pixeln, in der vier Vollfarben dargestellt werden können. In einer Zeile können dabei bis zu 85 Zeichen ausgegeben werden. Zur optimalen Ausnutzung der hohen Auflösung ist ein Monitor erforderlich.

Ein **PIXEL** ist der kleinste Punkt, der auf dem Bildschirm ausgegeben werden kann. In der hochauflösenden Betriebsart können in einer waagerechten Linie 512 Pixel dargestellt werden, in einer senkrechten 256 Pixel.

Größe und Form der Pixel sind in beiden Betriebsarten unterschiedlich. Trotzdem arbeitet ein QL-Programm, das zum Beispiel Kreise und Quadrate zeichnet, in beiden Betriebsarten korrekt und zeichnet nicht etwa in der einen Betriebsart Ellipsen und Rechtecke. Die Grafik-Prozeduren von SuperBASIC benutzen nämlich ein Koordinatensystem, das von den Pixeln unabhängig ist, das **grafische Koordinatensystem**.

Die senkrechte Achse des grafischen Koordinatensystems ist normalerweise in 100 Einheiten unterteilt. Sie werden später sehen, wie Sie den Maßstab und die Lage der Koordinatenachse verändern können (mit dem **SCALE**-Befehl).



Das grafische Koordinatensystem.

EINE FARBIGE LINIE

Hierzu müssen drei Dinge angegeben werden:

- **PAPER** (Hintergrundfarbe)
- **INK** (Schriftfarbe)
- **LINE** (Anfangs- und Endpunkte)

Mit dem folgenden Programm wird eine wie oben dargestellte Linie mit roter Farbe (Farbcode 2) auf einem weißen Hintergrund (Farbcode 7) gezeichnet.

```
NEW ⇐
100 PAPER 7 :CLS ⇐
110 INK 2 ⇐
120 LINE 50,60 TO 70,80 ⇐
RUN ⇐
```

In Zeile 100 wird zuerst die Hintergrundfarbe ausgewählt. Sie wird jedoch erst mit einem weiteren Befehl, wie beispielsweise **CLS**, wirksam. Mit ihm wird angegeben, daß der Bildschirm auf die aktuelle Hintergrund- oder Papierfarbe geändert wird.

BETRIEBSARTEN UND FARBEN

Bis jetzt spielt es noch keine Rolle, welche Bildschirm-Betriebsart benutzt wird. Die Betriebsart wirkt sich jedoch auf die Anzahl der darstellbaren Farben aus.

Mit MODE 8 können acht Grundfarben benutzt werden.
Mit MODE 4 können vier Grundfarben benutzt werden.

Die Farben werden mit den Codes der folgenden Tabelle angegeben.

Code	Farbe	
	MODE 8	MODE 4
0	Schwarz	Schwarz
1	Blau	Schwarz
2	Rot	Rot
3	Magenta	Rot
4	Grün	Grün
5	Zyan	Grün
6	Gelbe	Weiß
7	Weiß	Weiß

So erhält man beispielsweise bei **INK 3** mit **MODE 8** Magenta und mit **MODE 4** Rot.

In einem der späteren Abschnitte wird beschrieben, wie die Grundfarben auf verschiedene Art und Weise gemischt werden können, um ein recht breites Spektrum an Farben, Schattierungen und Mustern zu erzeugen.

AUSWIRKUNGEN DES ZUFALLS

Mit Zufallszahlen, die mit der **RND**-Funktion erzeugt werden können, können sehr interessante Effekte erzielt werden. So druckt beispielsweise:

```
PRINT RND(1 TO 6) ⇐
```

eine ganze Zahl im Bereich von 1 bis 6 aus. Diese Zahl wird zufällig ausgewählt, vergleichbar mit dem Werfen eines normalen sechsseitigen Würfels. Dies wird mit dem folgenden Programm verdeutlicht:

```
NEW ⇐
100 LET wurf = RND(1 TO 6) ⇐
110 PRINT wurf ⇐
RUN ⇐
```

Wird das Programm mehrmals ausgeführt, so erhalten Sie verschiedene Zahlen.

Sie können Zufallszahlen in jedem gewünschten Bereich erhalten. Sie erzeugt beispielsweise:

```
RND(0 TO 100)
```

eine ganze Zahl zwischen 0 und 100, die in der Grafik benutzt werden kann. Das Programm, mit dem eine Linie gezeichnet wird, kann so umgeschrieben werden, daß eine zufällige Farbe erzeugt wird. Wenn der Bereich der Zufallszahlen ab Null beginnt, kann die erste Zahl weggelassen und folgendes geschrieben werden:

```
RND(100)
NEW ⇐
100 PAPER 7: CLS ⇐
110 INK RND(5) ⇐
120 LINE 50,60 TO RND(100), RND(100) ⇐
RUN ⇐
```

Dadurch wird eine Zufallslinie erzeugt, die nahe der Bildschirmmitte beginnt und bei einem zufälligen Punkt beendet wird. Der Bereich der möglichen Farben hängt von der ausgewählten Betriebsart ab. Sie werden noch feststellen, daß es bei SuperBASIC oft Zahlenbereiche von 'beliebige Zahl **TO** beliebige Zahl' gibt.

UMRANDUNGEN

Der Teil des Bildschirms, in dem Linien gezeichnet und andere Ausgaben erstellt werden, wird als **Fenster** bezeichnet. Später wird beschrieben, wie die Größe und Position eines Fensters geändert oder andere Fenster erstellt werden können. Für

den Augenblick wollen wir uns damit zufriedengeben, einen Rand um das aktuelle Fenster zu zeichnen.

Sie können einen Rand um die innere Kante eines Fensters zeichnen, indem Sie beispielsweise:

```
BORDER 4,2 ⇐
```

eingeben. Dadurch wird eine Umrandung mit einer Breite von vier Pixeln in der roten Farbe (Code 2) gezeichnet. Die tatsächliche Größe des Fensters wird um die Umrandung verringert. Dies bedeutet, daß nachfolgend ausgedruckte Angaben oder Grafiken automatisch der neuen Fenstergröße angepaßt werden. Die einzige Ausnahme zu dieser Regel ist ein weiterer Rand, durch den der bestehende Rand ersetzt wird.

EINE EINFACHE SCHLEIFE

Computer können Operationen sehr schnell ausführen. Diese große Kapazität der Computer könnte jedoch nicht voll ausgeschöpft werden, wenn jeder Schritt als Befehl geschrieben werden müßte. Ein Vorarbeiter auf dem Bau hat dasselbe Problem. Möchte er, daß ein Arbeiter 100 Steine übereinandersetzt, so gibt er ihm nicht 100 separate Anweisungen. Er erteilt die Anweisung, eine bestimmte Aufgabe – Stein versetzen – 100mal auszuführen.

Bei BASIC wird bisher oft eine **GO TO**- (oder **GOTO**)-Anweisung benutzt, um eine Schleife oder Wiederholung zu erzeugen:

```
NEW ⇐
100 PAPER 6 : CLS ⇐
110 BORDER 1,2 ⇐
120 INK RND(5) ⇐
130 LINE 50,60 TO RND(100),RND(100) ⇐
140 GO TO 120 ⇐
RUN ⇐
```

Dieses Programm sollte besser nicht eingegeben werden, da es bei SuperBASIC bessere Möglichkeiten für Wiederholungen gibt. Dabei sollten bestimmte Dinge über den Programmaufbau berücksichtigt werden.

100 110	Fester Teil – wird nicht wiederholt
120 130	Veränderlicher Teil – wird wiederholt
140	Steuert den Programmfluß

Das obige Programm kann ohne die **GOTO**-Anweisung neu geschrieben werden. Statt dessen wird der zu wiederholende Teil zwischen **REPeat**- und **END REPeat**-Anweisungen gesetzt.

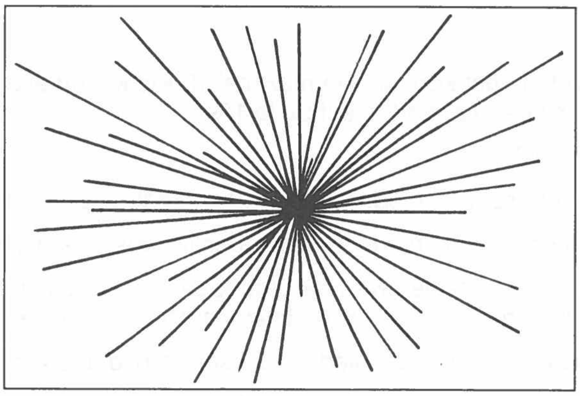
```
NEW ⇐
100 PAPER 6: CLS ⇐
110 BORDER 1,2 ⇐
120 REPeat stern ⇐
130   INK RND(5) ⇐
140   LINE 50,60 TO RND(100),RND(100) ⇐
150 END REPeat stern ⇐
RUN ⇐
```

Hier haben wir der *Wiederholstruktur* einen Namen gegeben, *Stern.* Die Struktur wird von den beiden folgenden Zeilen begrenzt:

REPeat stern
END REPeat stern

Die zwischen diesen beiden Anweisungen stehenden Angaben werden als **Inhalt** der Struktur bezeichnet. Mit den Großbuchstaben wird angegeben, daß **REP** eine zulässige Abkürzung für **REPeat** darstellt.

Mit diesem Programm werden fortwährend farbige Linien erzeugt, aus denen wie unten dargestellt ein Stern ensteht.



Das STERN-Programm

Durch Betätigung der Unterbrechungstasten kann dieses Programm gestoppt werden:

Halten Sie die [CTRL]-Taste gedrückt und betätigen Sie die [LEERTASTE].

SuperBASIC verfügt über eine vielseitige Methode für das Stoppen von Wiederholungen. Stellen Sie sich vor, daß innerhalb des Programms ständig dieselben Anweisungen ausgeführt werden. Wie kommt man aus dieser Schleife heraus? Mit einer **EXIT**-Anweisung. Zuerst muß es jedoch einen Grund geben, diese Schleife zu beenden. Sie können die Auswahl von Linienfarben erweitern, indem Sie als Programmänderung eingeben (geben Sie nicht **NEW** ein):

```
130 INK RND(0 to 6) ⇐
```

Erzeugt nun **RND** die Zahl 6, so wird die Schriftfarbe gleich wie die Hintergrundfarbe, und Sie sehen sie nicht.

Dies kann einen Grund für die Beendigung der Wiederholung darstellen. Das Programm kann nun folgendermaßen umgeschrieben werden:

```
NEW ⇐
100 PAPER 6: CLS ⇐
110 BORDER 1,2 ⇐
120 REPeat stern ⇐
130   LET farbe = RND(6) ⇐
140   IF farbe = 6 THEN EXIT stern ⇐
150   INK farbe ⇐
160   LINE 50,60 TO RND(100),RND(100) ⇐
170 END REPeat stern ⇐
RUN ⇐
```

Jetzt wird die Schleife solange wiederholt, bis *Farbe* den Wert 6 erhält. Dann wird die Schleife verlassen. Das Programm kann unmittelbar nach Zeile 170 fortgesetzt werden. Da hinter Zeile 170 keine weiteren Programmzeilen mehr vorhanden sind, stoppt das Programm.

Hier wurde ein weiteres wichtiges Konzept eingeführt, nämlich das Konzept der Entscheidung.

```
IF farbe = 6 THEN EXIT stern ⇐
```

Dies ist auch eine sehr wichtige Struktur, da sie eine Möglichkeit darstellt, zu entscheiden ob bestimmte Anweisungen ausgeführt werden sollen oder nicht. Sie wird als einfache binäre Entscheidung bezeichnet. Sie hat im allgemeinen folgende Form:

IF Bedingung THEN Anweisung(en)

Später wird noch deutlich, wie die beiden Konzepte der Wiederholung (oder Schleifenbildung) und der Entscheidung (oder Auswahl) die Hauptstrukturen für den Aufbau von Programmen darstellen.

TEST ZU KAPITEL 3

Bei dem folgenden Test können Sie maximal 12 Punkte erhalten. Vergleichen Sie Ihre Punktzahl mit den Antworten auf Seite 109.

1. Was ist ein Pixel?
2. Wie viele Pixel passen im niedrigauflösenden Modus über die Bildschirmbreite?
3. Wie viele Pixel passen im niedrigauflösenden Modus über die Bildschirmhöhe?
4. Mit welchen beiden Zahlen wird die 'Adresse' oder Position eines Punktes auf dem Bildschirm im grafischen Koordinatensystem angegeben?
5. Wie viele Farben stehen in der niedrigauflösenden Betriebsart zu verfügung?
6. Nennen Sie die Befehle, die folgende Funktionen ausführen:

   a) Zeichnen von Linien
   b) Auswahl einer Zeichenfarbe
   c) Auswahl einer Hintergrundfarbe
   d) Zeichnen einer Umrandung (vier Punkte)
7. Wie heißen die Anweisungen, mit denen die **REPeat**-Schleife geöffnet und geschlossen wird?
8. Wie kann man eine **REPeat**-Schleife verlassen?
9. Warum haben Schleifen in SuperBASIC Namen?

AUFGABEN ZU KAPITEL 3

1. Schreiben Sie ein Programm, mit dem gerade Linien über den Bildschirm gezeichnet werden. Diese Linien sollen eine zufällige Länge und Richtung aufweisen. Jede Linie soll an dem Punkt beginnen, an dem die vorhergehende Linie beendet wurde. Außerdem soll für jede Linie eine zufällige Farbe ausgewählt werden.
2. Schreiben Sie ein Programm, mit dem Zufallslinien gezeichnet werden, wobei jede Linie an einem zufälligen Punkt am linken Rand des Bildschirms beginnen soll.
3. Schreiben Sie ein Programm, mit dem Zufallslinien gezeichnet werden, wobei jedoch alle Linien an demselben Punkt in der unteren Ecke des Bildschirms beginnen sollen.
4. Schreiben Sie ein Programm, mit dem Linien mit zufälliger Länge, zufälligen Startpunkten und zufälliger Farbe gezeichnet werden. Alle Linien sollen horizontal verlaufen.
5. Wie bei Aufgabe 4, die Linien sollen jedoch vertikal verlaufen.
6. Schreiben Sie ein Programm, mit dem eine quadratische "Spirale" so gezeichnet wird, daß jede Linie eine zufällige Farbe aufweist.

HINWEIS: Suchen Sie zuerst nach den Koordinaten einiger der Ecken und setzen Sie sie in Vierergruppen zusammen. Dadurch ergibt sich ein Muster.

KAPITEL 4

ZEICHEN UND STRINGS

Lehrer möchten manchmal einschätzen, ob ihre Schüler schon für die Lektüre bestimmter Bücher oder Stoffe bereit sind. Hierzu werden verschiedene Tests benutzt. Bei einigen dieser Tests wird die durchschnittliche Länge von Wörtern und Sätzen berechnet. Wir werden die Bearbeitung von Wörter oder **Zeichen-Strings** vorstellen, indem wir einfache Lösungen für die Ermittlung der durchschnittlichen Wortlänge untersuchen.

Wir sprechen von Folgen aus Buchstaben, Ziffern oder anderen Symbolen, bei denen es sich um Wörter handeln kann oder auch nicht. Deshalb wurde der Ausdruck 'Zeichen-String' erfunden. Er wird normalerweise nur **String** genannt. Strings werden ähnlich wie Zahlen verarbeitet, allerdings führen wir natürlich nicht dieselben Operationen mit ihnen aus. Strings werden nicht multipliziert oder subtrahiert. Sie werden aneinandergefügt, getrennt, durchsucht und nach Bedarf verarbeitet.

NAMEN UND ABLAGEFÄCHER FÜR STRINGS

Sie können Ablagefächer für Strings erstellen. Sie können Zeichen-Strings in Ablagefächern speichern und die Informationen genau wie Zahlen benutzen. Möchten Sie (bitte nicht alle gleichzeitig) Wörter wie:

ERSTER ZWEITER DRITTER
und
JANUAR FEBRUAR MÄRZ

speichern, so können Sie zwei Ablagefächer benennen:



Beachten Sie das Dollarzeichen. Ablagefächer für Strings unterscheiden sich intern von Ablagefächern für Zahlen. SuperBASIC muß nun wissen, welche Art von Information ein Ablagefach enthält – eine Zahl oder einen String. Die Namen der Ablagefächer für Strings müssen deshalb mit $ beendet werden. Ansonsten sind die Regeln für die Auswahl von Namen dieselben wie bei den Ablagefächern für Zahlen.

Sie können:

wochentag$ als wochentagdollar
monat$ als monatdollar

aussprechen.

Die **LET**-Anweisung wird wie bei Zahlen benutzt. Geben Sie:

```
LET wochentag$ = "ERSTER" ←
```

ein, so wird ein internes Ablagefach namens *wochentag$*, mit dem Inhalt ERSTER erstellt:



Die Anführungszeichen werden nicht gespeichert. Sie werden in der **LET**-Anweisung Ablagefaches angezeigt werden. Sie können dies überprüfen mit:

```
PRINT wochentag$ ←
```

Die Ausgabe am Bildschirm zeigt nun den Inhalt des Ablagefaches

```
ERSTER
```

Anstelle der Anführungszeichen können auch Apostrophe benutzt werden.

LÄNGE VON STRINGS

Bei SuperBASIC kann die Länge oder Anzahl von Zeichen eines Strings problemlos ermittelt werden. So geben Sie beispielsweise einfach:

```
PRINT LEN(wochentag$) ⏎
```

ein. Enthält das Ablagefach, *wochentag$*, ERSTER, so wird die Zahl 6 angezeigt. Die Auswirkungen werden anhand eines einfachen Programms deutlich:

```
NEW ⏎
100 LET wochentag$ = ''ERSTER'' ⏎
110 PRINT LEN(wochentag$) ⏎
RUN ⏎
```

Auf dem Bildschirm muß:

```
6
```

angezeigt werden. **LEN** ist ein SuperBASIC-Befehl.

Bei einer anderen Methode, mit der dasselbe Ergebnis erzielt wird, wird sowohl ein Ablagefach für einen String als auch ein Ablagefach für eine Zahl benutzt.

```
NEW ⏎
100 LET wochentag$ = ''ERSTER'' ⏎
110 LET lang = LEN(wochentag$) ⏎
120 PRINT lang ⏎
RUN ⏎
```

Auf dem Bildschirm muß:

```
6
```

wie vorher angezeigt werden. Die beiden internen Ablagefächer enthalten folgende Werte:

Wochentag$ [ERSTER]   lang [6]

Nun wollen wir uns wieder dem Problem der durchschnittlichen Länge von Wörtern zuwenden.

Schreiben Sie ein Programm, um die durchschnittliche Länge der drei folgenden Wörter zu ermitteln:

ERSTER APRIL 1985

PROGRAMM-AUFBAU

Sobald die Probleme über das Normalmaß hinausgehen, wird empfohlen, den **Programmaufbau** zu skizzieren, bevor das Programm selbst geschrieben wird.

1. Die drei Wörter werden in Ablagefächern gespeichert.
2. Die Länge der Wörter wird berechnet und gespeichert.
3. Der Durchschnittswert wird berechnet.
4. Das Ergebnis wird berechnet.

```
NEW ⏎
100 LET wochentag$ = ''ERSTER'' ⏎
110 LET monat$ = ''APRIL'' ⏎
120 LET jahr$ = ''1985'' ⏎
130 LET lang1 = LEN(wochentag$) ⏎
140 LET lang2 = LEN(monat$) ⏎
150 LET lang3 = LEN(jahr$) ⏎
160 LET summe = lang1 + lang2 + lang3 ⏎
170 LET durchschnitt = summe/3⏎
180 PRINT durchschnitt ⏎
RUN ⏎
```

Das Symbol **/** bedeutet **dividiert durch.** Das Ergebnis der Ausführung dieses Programms ist einfach die folgende Ausgabe:

```
5
```

Hier wurden nun acht interen Ablagefächer benutzt:

wochentag$	ERSTER	lang1	5
monat$	APRIL	lang2	2
jahr$	1985	lang3	8
		Summe	15
		Durchschnitt	5

Sind Sie der Auffassung, daß dies ein recht großer Aufwand für eine so einfache Aufgabe ist, so können Sie dies sicherlich kürzen. Die kürzeste Version besteht in einer einzigen Zeile, die jedoch schwieriger zu lesen ist. Bei einem vernünftigen Kompromiß wird das Symbol **&** benutzt, das folgende Operation bwirkt:

Verbinden von zwei Strings

Nun nehmen Sie folgende Eingabe vor:

```
NEW ←
100 LET wochentag$ = ''ERSTER'' ←
110 LET monat$ = ''APRIL'' ←
120 LET jahr$ = ''1985'' ←
130 LET satz$ = wochentag$ & monat$ & jahr$ ←
140 LET lang = LEN(satz$) ←
150 PRINT lang/3 ←
RUN ←
```

Wie vorher wird 5 ausgegeben. Es gibt jedoch eine Reihe anderer interner Auswirkungen:

wochentag$	ERSTER	Länge	15
monat$	APRIL		
jahr$	1985		
begriff$	ERSTERAPRIL1985		

Es gibt noch eine vernünftigere Vereinfachung, bei der **READ** und **DATA** anstelle der drei ersten **LET**-Anweisungen benutzt wird. Nehmen Sie folgende Eingabe vor:

```
NEW ←
100 READ wochentag$, monat$, jahr$ ←
110 LET satz$ = wochentag$ & monat$ & jahr$ ←
120 LET lang = LEN(satz$) ←
130 PRINT lang/3 ←
140 DATA ''ERSTER'',''APRIL'',''1985'' ←
RUN ←
```

Intern wirkt sich diese Version genau wie die vorhergehende Version aus. Mit **READ** werden interne Ablagefächer mit entsprechenden Werten wie bei **LET** erstellt.

NAMEN UND STRING-VARIABLEN

Namen von Ablagefächern, wie beispielsweise:

wochentag$
monat$
jahr$
begriff$

werden als String-Namen bezeichnet. Mit den Dollarzeichen wird einfach angegeben, daß die Ablagefächer für Zeichen-Strings benutzt werden. Das Dollarzeichen muß stets am Ende stehen.

Ablagefächer dieser Art werden als **String-Variable** bezeichnet, da sie nur Zeichen-Strings enthalten können, die sich während der Ausführung eines Programms ändern können.

Der Inhalt derartiger Ablagefächer wird als Wert bezeichnet. So können Wörter wie 'ERSTER' und 'APRIL' Werte von String-Variablen namens *wochentag$* und *monat$* sein.

ZUFÄLLIGE ZEICHEN

Zur Erzeugung zufälliger Buchstaben können Sie Zeichencodes benutzen (siehe *Kapitel Begriffe*). Die Großbuchstaben A bis Z haben die Codes 65 bis 90. Mit der Funktion **CHR$** werden diese Codes in Buchstaben umgesetzt. Mit dem folgenden Programm wird ein Buchstabe B ausgedruckt.

```
NEW ⇐
100 LET zeichencode = 66 ⇐
110 PRINT CHR$(zeichencode) ⇐
RUN ⇐
```

Das folgende Programm erzeugt immer wieder die Buchstaben A, B oder C, bis zufällig die Buchstabenfolge "ABC" entsteht.

```
NEW ⇐
100 REPeat alphabet ⇐
110   LET erster$ = CHR$(RND(65 TO 67)) ⇐
120   LET zweiter$ = CHR$(RND(65 TO 67)) ⇐
130   LET dritter$ = CHR$(RND(65 TO 67)) ⇐
140   LET reihenfolge$ = erster$ & zweiter$ & dritter$ ⇐
150   PRINT ! reihenfolge$ ⇐
160 IF reihenfolge$ = ''ABC'' THEN EXIT alphabet ⇐
170 END REPeat alphabet ⇐
RUN ⇐
```

Zufällige Zeichen, wie Zufallszahlen oder zufällige Punkte sind beim Erlernen der Programmierung besonders nützlich. Mit ihnen können problemlos interessante Effekte für Programmbeispiele und Übungen erzielt werden.

Beachten Sie, welche Auswirkungen die !...! auf den Abstand in der Ausgabe haben.

TEST ZU KAPITEL 4

Bei dem folgenden Test können Sie maximal 10 Punkte erzielen. Vergleichen Sie Ihre Punktzahl mit den Anworten auf Seite 109.

1. Was ist ein Zeichen-String?
2. Wie lautet die übliche Abkürzung für den Ausdruck 'Zeichen-String'?
3. Was unterscheidet den Namen einer String-Variablen von dem einer numerischen Variablen?
4. Wie kann ein Name wie "wort$" noch ausgesprochen werden?
5. Mit welchem Befehl wird die Anzahl von Zeichen in einem String ermittelt?
6. Mit welchem Symbol werden zwei Strings miteinander verknüpft?
7. Leerzeichen können Teil eines Strings sein. Wie werden die Grenzen eines Strings markiert?
8. Werden die Anführungszeichen bei der Ausführung einer Anweisung wie:

   LET fleisch$ = "Schnitzel"

   gespeichert?
9. Mit welcher Funktion wird eine entsprechende Codenummer in einen Buchstaben umgewandelt?
10. Wie können zufällige Großbuchstaben erzeugt werden?

AUFGABEN ZU KAPITEL 4

1. Speichern Sie die Wörter "Guten" und "Tag" in zwei separaten Variablen. Mit einer **LET**-Anweisung verknüpften Sie die Werte der beiden Variablen zu einer dritten Variablen. Drucken Sie das Ergebnis aus.
2. Speichern Sie die folgenden Wörter in drei separaten Ablagefächern:

   Es werde licht

   Verbinden Sie die Wörter zu einem Satz, wobei Sie Leerzeichen und einen Punkt hinzufügen. Speichern Sie den ganzen Satz in einer Variablen, *satz$*, und drucken Sie ihn, sowie die Gesamtzahl der in ihm enthaltenen Zeichen aus.
3. Schreiben Sie ein Programm mit den Befehlen:

   ```
   CHR$(RND(65 TO 90)) ⇐
   ```

   um 100 aus drei zufälligen Buchstaben bestehende Wörter zu bilden. Prüfen Sie, ob Sie zufällig richtige deutsche Worte gebildet haben. Testen Sie die Auswirkungen von:

   a) ; am Ende einer **PRINT**-Anweisung.
   b) ! rechts oder links neben einem auszudruckenden Element.

KAPITEL 5
PRAKTISCHE ERFAHRUNGEN

Dabei haben Sie sicher festgestellt, daß sich die folgenden Praktiken als hilfreich erweisen:

1. Benutzung von Kleinbuchstaben für Namen: Namen von Variablen (Ablagefächern) oder Wiederholungsstrukturen, usw.
2. Einrücken von Anweisungen, um die Struktur von Schleifen sichbar zu machen.
3. Sorgfältig ausgewählte Namen, mit denen die Bedeutung einer Variablen oder Wiederholstruktur erkennbar wird.
4. Bearbeitung eines Programms durch:
   - Ersetzen einer Zeile
   - Einfügen einer Zeile
   - Löschen einer Zeile.

PROGRAMME ALS BEISPIELE

Sie haben nun einen Punkt erreicht, an dem Sie Programme studieren können, um aus ihnen zu lernen und zu verstehen, wie sie arbeiten. So wissen Sie mittlerweile auch genau, wie Programme ausgeführt werden. Deshalb werden wir in den folgenden Abschnitten auf die ständige Wiederholung von:

`NEW`	vor jedem Programm
⬅	am Ende jeder Zeile
`RUN`	für den Start eines Programms

verzichten. Sie wissen, daß Sie all diese Funktionen benutzen müssen, wenn Sie ein Programm eingeben und ausführen möchten. Werden Sie in dem Text jetzt weggelassen, so sehen Sie die anderen Details deutlicher, wenn Sie sich vorstellen, was das Programm bei der Ausführung tut.

Werden diese Angaben weggelassen, so können Programme ohne die technischen Details problemloser benutzt und verstanden werden. So erzeugt beispielsweise das folgende Programm zufällige Großbuchstaben, bis ein Z angezeigt wird.

```
100 REPeat alphabet
110   LET zeichencode = RND(65 TO 90)
120   buchstabe$ = CHR$(zeichencode)
130   PRINT buchstabe$
140   IF buchstabe$ = ''Z'' THEN EXIT alphabet
150 END REPeat alphabet
```

In diesem und den folgenden Kapiteln werden die Programme ohne die Symbole für **ENTER** angegeben. Direkte Befehle werden ebenfalls ohne die **ENTER**-Symbole beginnen. Dennoch müssen Sie diese Tasten wie üblich benutzen, außerdem müssen Sie daran denken, **NEW** und **RUN** nach Bedarf zu benutzen

AUTOMATISCHE ZEILEN-NUMERIERUNG

Es ist mühsam, Zeilennummern manuell einzugeben. Stattdessen können Sie:

```
AUTO
```

eingeben, bevor Sie mit der Programmierung beginnen. In diesem Fall antwortet der QL mit einer Zeilennummer:

```
100
```

Nun geben Sie weiter Programmzeilen ein, bis das Programm beendet ist, wobei folgendes auf dem Bildschirm angezeigt wird:

```
100 PRINT ''Erste''
110 PRINT ''Zweite''
120 PRINT ''Ende''
```

Um die automatische Erzeugung von Zeilennummern zu beenden, benutzen Sie die Tastenkombination für die **BREAK**-Funktion:

Halten Sie die **CTRL**-Taste gedrückt und betätigen Sie die **LEERTASTE**. Dadurch wird die Meldung:

```
130 ABGEBROCHEN
```

angezeigt.

Zeile 130 ist in dem Programm nicht enthalten.

Kommt es zu einem Fehler, der nicht zu einer Unterbrechung der automatischen Zeilennumerierung führt, so kann mit der Eingabe fortgefahren und die Zeile später mit **EDIT** bearbeitet werden. Soll mit einer bestimmten Zeilennummer begonnen werden, beispielsweise mit Zeile 600, und sollen die Zeilen nicht in 10er Schritten numeriert werden, so können Sie beispielsweise eine Numerierung in 5er Schritten eingeben:

```
AUTO 600,5
```

Danach werden die Zeilen mit 600, 605, 610 usw. numeriert.

Um **AUTO** rückgängig zu machen, betätigen Sie **CTRL** und die **Leertaste** gleichzeitig.

BEARBEITEN EINER ZEILE

Um eine Zeile zu bearbeiten, geben Sie einfach **EDIT** gefolgt von der Zeilennummer ein. Zum Beispiel:

```
EDIT 110
```

Danach wird die Zeile angezeigt, wobei der Cursor am Zeilenende steht:

```
110 PRINT ''Zweite''
```

Der Cursor kann folgendermaßen bewegt werden:

← bewegt den Cursor um eine Stelle nach links
→ bewegt den Cursor um eine Stelle nach rechts

Um ein links davon stehendes Zeichen zu löschen, betätigen Sie:

CTRL zusammen mit ←

Um das Zeichen an der Cursor-Position zu löschen, betätigen Sie

CTRL zusammen mit →.

Das rechts vom Cursor stehende Zeichen rückt auf, um die Lücke zu schließen.

BENUTZUNG VON MICRODRIVE-KASSETTEN

Bevor eine neue Microdrive-Kassette benutzt wird, muß sie formatiert werden. Hierzu wird wie in der *Einführung* beschrieben, vorgegangen. Die Auswahl des Namens für die Kassette unterliegt denselben Regeln wie die SuperBASIC-Namen, allerdings sind die Namen für Kassetten auf 10 Zeichen beschränkt. Es wird empfohlen, den Namen der Kassette mit den selbstklebenden Etiketten auf die Kassette zu schreiben. Sie sollten stets eine Sicherungskopie der Programme oder Daten anfertigen. Hierzu wird wie in dem Kapitel "Information" des Anwenderhandbuchs beschrieben vorgegangen.

> ACHTUNG:
> Wird eine Kassette formatiert, auf der Programme oder Daten stehen, so sind SÄMTLICHE Programme oder Daten verloren.

SICHERN VON PROGRAMMEN

Mit dem folgenden Programm werden Umrandungen mit einer Breite von acht Pixeln in roter Farbe (Code 2) in drei Fenstern gezeichnet, die mit #0, #1, #2 bezeichnet sind.

```
100 REMark Umrandung
110 FOR k = 0 TO 2 : BORDER #k,8,2
```

Dieses Programm kann in einem Microdrive gesichert werden, indem eine Kassette eingelegt und:

```
SAVE mdv1_Umrandung
```

eingegeben wird. Das Programm wird in einer Microdrive-Datei namens **umrandung** gesichert.

Wenn Sie mit dem SAVE-Befehl ein Programm auf Microdrive-Kassette sichern, so legt der QL eine sogenannte **Datei** an, in die er das Programm schreibt. Unter dem Namen, den Sie mit dem **SAVE**-Befehl vergeben haben, kann der QL die Datei unter den verschiedenen Informationen, die auf der Kassette gespeichert sind, jederzeit wieder finden. Ähnlich wie Programme können auch Daten auf einer Microdrive-Kassette dauerhaft gespeichert werden.

ÜBERPRÜFEN EINER KASSETTE

Möchten Sie wissen, welche Programme oder Datendateien auf einer bestimmten Kassette stehen, so legen Sie diese in Microdrive 1 ein und geben:

```
DIR mdv1_
```

ein. Das Inhaltsverzeichnis wird auf dem Bildschirm angezeigt. Ist die Kassette in Microdrive 2 eingelegt, so geben Sie statt dessen:

```
DIR mdv2_
```

ein.

KOPIEREN VON PROGRAMMEN UND DATEIEN

Sobald ein Programm als Datei auf einer Microdrive-Kassette gespeichert wurde, kann es in andere Dateien kopiert werden. Dies ist eine Möglichkeit zur Anfertigung von Sicherungskopien einer Microdrive-Kassette. Sie können alle alten Programme auf eine andere Kassette in Microdrive 2 kopieren, indem Sie:

```
COPY mdv1_Umrandung TO mdv2_Umrandung
```

eingeben.

LÖSCHEN EINER KASSETTEN-DATEI

Um eine Datei namens *prog* zu löschen, geben Sie:

```
DELETE mdv1_prog
```

ein.

LADEN VON PROGRAMMEN

Ein Programm kann von einer Microdrive-Kassette geladen werden, indem:

```
LOAD mdv2_Umrandung
```

eingegeben wird. Sie können das Programm testen, indem Sie es mit:

LIST auflisten,
RUN ausführen.

Anstatt **LOAD** gefolgt von **RUN** zu benutzen, können Sie die beiden Operationen in einem Befehl kombinieren.

```
LRUN mdv2_Umrandung
```

Das Programm wird geladen und sofort ausgeführt.

MISCHEN VON PROGRAMMEN

Hier wollen wir davon ausgehen, daß Sie zwei Programme in Microdrive 1 als *programm1* und *programm2* gesichert haben.

```
100 PRINT ''Programm 1''
110 PRINT ''Programm 2''
```

Geben die nun:

```
LOAD mdv1_prog1
```

gefolgt von:

```
MERGE mdv1_prog2
```

ein, so werden die beiden Programme zu einem Programm gemischt. Um dies zu überprüfen, geben Sie **LIST** ein. Auf dem Bildschirm muß folgendes angezeigt werden:

```
100 PRINT ''Programm 1''
110 PRINT ''Programm 2''
```

Wird ein Programm mit **MERGE**gemischt, so müssen Sie prüfen, ob sich seine Zeilennummern von den Zeilennummern des schon im Hauptspeicher stehenden Programms unterscheiden. Ansonsten werden einige der Zeilen des ersten Programms überschrieben. Diese Funktion wird sehr nützlich, sobald Sie mit der Handhabung von Prozeduren vertraut sind. Dann wird es ganz normal, ein Programm zu erstellen, indem Prozeduren oder Funktionen zu ihm hinzugefügt werden.

ALLGEMEINE HINWEISE

Bei den Kassetten muß mit Sorgfalt und Methode vorgegangen werden. Fertigen Sie stets eine Sicherungskopie an. Befürchten Sie ein Problem bei einer Kassette oder einem Microdrive, so fertigen Sie eine zweite Sicherungskopie an. Computer-Profis

verlieren nur ganz selten Daten. Sie wissen, daß selbst die besten Computer oder Einheiten gelegentlich Fehler aufweisen und berücksichtigen dies schon im voraus. Soll ein Programm mit einem bestimmten Namen aufgerufen werden, wie beispielsweise *Quadrat*, so wird empfohlen, Namen wie *qdr1, qdr2*... für die vorläufigen Versionen zu benutzen. Sobald das Programm seine endgültige Form aufweist, fertigen Sie mindestens zwei Kopien namens *Quadrat* an. Die anderen Kopien können dann durch Neuformatieren der Kassette oder mit dem **DELETE**-Befehl gelöscht werden.

TEST ZU KAPITEL 5

Bei dem folgenden Test können Sie maximal 13 Punkte erhalten. Vergleichen Sie Ihr Testergebnis mit den Antworten auf Seite 110.

1. Warum ist es zweckmäßig, in einem Programm Namen, die Sie selbst auswählen, in Kleinbuchstaben zu schreiben?
2. Was ist der Zweck der Einrückung von Programmzeilen?
3. Anhand welcher Richtlinien wählen Sie normalerweise die Namen für Variablen und Schleifen aus?
4. Geben Sie drei Arten an, mit denen ein Programm im Hauptspeicher des Computers bearbeitet werden kann (drei Punkte).
5. Woran muß bei Beendigung jedes Befehls oder jeder Programmzeile nach der Eingabe gedacht werden?
6. Was müssen Sie normalerweise eingeben, bevor Sie ein Programm über die Tastatur eingeben?
7. Was muß am Anfang jeder als Teil eines Programms zu speichernden Zeile stehen?
8. Was müssen Sie eingeben, um ein Programm auszuführen?
9. Mit welchem Befehl können Sie Angaben in ein Programm eingeben, die sich nicht auf die Ausführung auswirken?
10. Mit welchen beiden Befehlen können Sie Programme auf Kassetten speichern und wieder von Kassetten aufrufen? (Zwei Punkte)

AUFGABEN ZU KAPITEL 5

1. Schreiben Sie das folgende Programm neu, wobei Sie Kleinbuchstaben benutzen, um die Darstellung zu verbessern. Fügen Sie die Wörter **NEW** und **RUN** hinzu. Benutzen Sie Zeilennummern und das Symbol für **ENTER**, genau wie bei der Eingabe und Ausführung eines Programms. Mit **REMark** geben Sie dem Programm einen Namen.

```
LET ZWEI$ = ''ZWEI''
LET VIER$ = ''VIER''
LET SECHS$ = ZWEI$ & VIER$
PRINT LEN(sechs$)
```

   Erklären Sie, warum zwei und vier 8 ergeben kann.

2. Benutzen Sie die Einrückung, Kleinbuchstaben, **NEW**, **RUN**, Zeilennummern und das Symbol für **ENTER**, um zu zeigen, wie Sie das folgende Programm wirklich eingeben und ausführen würden:

```
REPEAT LOOP
LETTER_CODE = RND(65 TO 90)
LET LETTER$ = CHR$(LETTER_CODE)
PRINT LETTER$
IF LETTER$ = 'Z' THEN EXIT LOOP
END REPEAT LOOP
```

3. Schreiben Sie das folgende Programm in einem besseren Stil mit sinvollen Variablennamen und einer besseren Darstellung neu. Schreiben Sie das Programm so, wie Sie es eingeben würden:

```
LET S = 0
REPeat TOTAL
LET N = RND(1 TO 6)
PRINT ! N !
LET S = S + N
IF n = 6 THEN EXIT TOTAL
END REPeat TOTAL
PRINT S
```

Entscheiden Sie, was das Programm tut. Danach geben Sie es ein und führen es aus, um zu sehen, ob Ihre Entscheidung richtig war.

KAPITEL 6
TABELLEN UND FOR-SCHLEIFEN

WAS IST EINE TABELLE

Sie wissen schon, daß die Werte von Variablen Zahlen- oder Zeichen-Strings sein können. Sie können sich diese als Zahlen oder Wörter vorstellen, die in internen Ablagefächern gespeichert werden. Wir wollen hier von vier Hobbygärtnern ausgehen, die jeweils eine besondere Blumenart züchten. Die Namen der Blumen enden alle mit einem Dollar-Symbol.

Rose$ Heide$ Nelke$ Tulpe$

Die vier Hobbygärtner haben folgende Namen:

UDO	FRI	HEI	KAR

Sie können nun mit zwei Methoden den von ihnen gezüchteten Blumen zugeordnet werden:

Programm 1

```
100 LET Rose$ = ''UDO''
110 LET Heide$ = ''FRI''
120 LET Nelke$ = ''HEI''
130 LET Tulpe$ = ''KAR''
140 PRINT ! rose$ ! heide$ ! nelke$ ! tulpe$
```

Programm 2

```
100 READ rose$, heide$, nelke$, tulpe$
110 PRINT ! rose$ ! heide$ ! nelke$ ! tulpe$
120 DATA ''UDO'', ''FRI'', ''HEI'', ''KAR''
```

rose$	heide$	nelke$	tulpe$
↓	↓	↓	↓
UDO	FRI	HEI	KAR

Je größer die Datenmengen werden, desto größer werden die Vorteile von **READ** und **DATA** gegenüber **LET**. Sobald die Daten jedoch wirklich umfangreich werden, wird es schwierig, die Namen der Blumen zu finden.

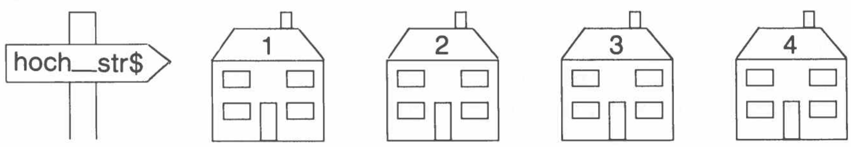
Dieses und viele andere Probleme bei der Datenverarbeitung kann mit einer neuen Art von Ablagefach oder Variablen gelöst werden, bei der mehrere Daten denselben Namen benutzen. Sie müssen sich jedoch unterscheiden, so daß jede Variable auch eine Nummer, wie beispielsweise Hausnummern in derselben Straße hat. Angenommen, es werden vier Häuser in der Hoch-Straße mit den Namen 1 bis 4 gesucht. Bei SuperBASIC sagen wir, daß es sich um eine **Tabelle** mit vier Elementen handelt. Die Tabelle hat den Namen *hoch_str$*. Die vier Häuser haben die Nummern 1 bis 4.

Diese Tabellenvariablen können jedoch nicht wie normale (einfache) Variablen benutzt werden. Zuerst müssen die Dimensionen (d.h. die Größe) der Tabelle deklariert werden. Der Computer weist den Platz intern zu und muß wissen, wie viele String-Variablen in der Tabelle vorhanden sind. Außerdem muß er die Maximallänge jeder String-Variablen kennen. Hierzu wird eine **DIM**-Anweisung benutzt:

DIM hoch_str$ (4,3)

- 3 — **Maximale Länge der Strings**
- 4 — **Anzahl von String-Variablen**

Nachdem die **DIM**-Anweisung ausgeführt wurde, können die Variablen benutzt werden. Die vier "Häuser" benutzen einen gemeinsamen Namen, *hoch_str$*. Jedes Haus verfügt jedoch über eine eigene Nummer und kann bis zu drei Zeichen enthalten:



Die folgenden fünf Programme tun alle dasselbe: sie führen dazu, daß die vier Häuser 'belegt' werden. Mit **PRINT** wird dann gezeigt, daß die 'Belegung' tatsächlich durchgeführt wurde. Bei der letzten Methode werden nur vier Zeilen benutzt. Die vier anderen Methoden führen jedoch so zu dieser letzten Methode, daß von schon bekannten Ideen zu neuen Ideen und neuen Anwendungen der alten Ideen ausgegangen wird. Dies führt letztendlich auch zu größerer Wirtschaftlichkeit.

Haben Sie die beiden ersten oder drei ersten Methoden verstanden, so können Sie direkt zu den Methoden 4 und 5 gehen.

Sollten Sie jedoch noch Zeit haben, so können diese mit den Methoden 1, 2 und 3 beseitigt werden.

Programm 1

```
100 DIM hoch_str$(4,3)
110 LET hoch_str$(1) = ''UDO''
120 LET hoch_str$(2) = ''FRI''
130 LET hoch_str$(3) = ''HEI''
140 LET hoch_str$(4) = ''KAR''
150 PRINT ! hoch_str$(1) ! hoch_str$(2) !
160 PRINT ! hoch_str$(3) ! hoch_str$(4) !
```

Programm 2

```
100 DIM hoch_str$(4,3)
110 READ hoch_str$(1), hoch_str$(2), hoch_str$(3),
        hoch_str$(4)
120 PRINT ! hoch_str$(1) ! hoch_str$(2) !
130 PRINT ! hoch_str$(3) ! hoch_str$(4)
140 DATA ''UDO'', ''FRI'', ''HEI'', ''KAR''
```

Mit diesem Programm wird dargelegt, wie Variablennamen sparsam eingesetzt werden. Die ständige Wiederholung von *hoch_str$* ist jedoch nicht nur zeitaufwendig, sondern gestaltet die Programme auch unübersichtlich. Wir können hier wieder eine bekannte Technik – die **REPeat**-Schleife benutzen, um die Dinge noch weiter zu verbessern. Hier wird ein Zähler, *Zahl* benutzt, der bei jedem Durchlauf der **REPeat**-Schleife um 1 erhöht wird.

Programm 3

```
100 RESTORE 190
110 DIM hoch_str$(4,3)
120 LET nummer = 0
130 REPeat häuser
140   LET nummer = nummer + 1
150   READ hoch_str$(nummer)
160   IF nummer = 4 THEN EXIT häuser
170 END REPeat häuser
180 PRINT hoch_str$(1)! hoch_str$(2)! hoch_str$(3)!
         hoch_str$(4)
140 DATA ''UDO'', ''FRI'', ''HEI'', ''KAR''
```

Eine spezielle Schleife, bei der eine bestimmte Operation eine bestimmte Anzahl von Malen ausgeführt werden muß, heißt **FOR**-Schleife. In einer derartigen Schleife wird automatisch von 1 bis 4 gezählt. Sie wird also beendet, sobald alle vier Elemente der Tabelle verarbeitet wurden.

Programm 4

```
100 RESTORE 160
110 DIM hoch_str$(4,3)
120 FOR nummer = 1 TO 4
130   READ hoch_str$(nummer)
140   PRINT ! hoch_str$(nummer) !
150 END FOR nummer
160 DATA ''UDO'', ''FRI'', ''HEI'', ''KAR''
```

Die Ausgabe aus allen vier Programmen ist dieselbe:

```
UDO FRI HEI KAR
```

Damit wird bewiesen, daß die Daten richtig intern in den vier Tabellen-Variablen gespeichert wurden:

hoch_str$	UDO	FRI	HEI	KAR
	1	2	3	4

Methode 4 ist bis jetzt eindeutig die beste, da sie 4, 40 oder 400 Elemente genau gleich verarbeiten kann, indem Sie einfach die Zahl 4 ändert und weitere **DATA**-Elemente hinzufügen. Sie können beliebig viele **DATA**-Anweisungen benutzen.

In ihrer einfachsten Form entspricht die **FOR**-Schleife der einfachsten Form der **REPeat**-Schleife. Die beiden Schleifen können miteinander verglichen werden:

```
100 REPeat empfang
110  PRINT ''Guten Tag''
120 END REPeat empfang
```

```
100 FOR empfang = 1 TO 40
110  PRINT ''Guten Tag''
120 END FOR empfang
```

Beide Schleifen können ausgeführt werden. Mit der **REPeat**-Schleife wird "Guten Tag"endlos ausdgedruckt (dieses Ausdrucken kann mit der **BREAK**-Funktion gestoppt werden), während "Guten Tag" mit der **FOR**-Schleife nur 40mal ausgedruckt wird.

Hier wird darauf hingewiesen, daß der Name der **FOR**-Schleife auch eine Variable ist, *empfang*, deren Wert sich im Verlauf der Programmausführung von 1 bis 40 ändert. Diese Variable wird auch gelegentlich als **Schleifen-Variable** oder **Steuervariable** der Schleife bezeichnet.

Beachten Sie, daß die Struktur beider Schleifen folgende Form aufweist:

Öffnende Anweisung
  Inhalt
Schließende Anweisung

Bestimmte Strukturen verfügen jedoch über zulässige Kurzformen, die benutzt werden, wenn der Inhalt der Schleife nur eine oder wenige Anweisungen umfaßt. Bei der **FOR**-Schleife sind Kurzformen zulässig. Deshalb können wir das Programm in der wirtschaftlichsten Form überhaupt schreiben.

Programm 5

```
100 RESTORE 140 : CLS
110 DIM hoch_str$(4,3)
120 FOR nummer = 1 TO 4 : READ hoch_str$(nummer)
130 FOR nummer = 1 TO 4 : PRINT ! hoch_str$(nummer) !
140 DATA ''UDO'', ''FRI'', ''HEI'', ''KAR''
```

Die Doppelpunkte werden benutzt, um die einzelnen Anweisungen einer Zeile voneinander zu trennen. Wenn eine **FOR**-Schleife nur eine Zeile umfaßt, braucht sie nicht mit **END FOR** abgeschlossen zu werden.

Es gibt sogar noch eine kürzere Form, das obige Programm zu schreiben. Um den Inhalt der Tabelle *hoch_str$* auszudrucken, kann Zeile 130 durch:

```
130 PRINT ! hoch_str$ !
```

ersetzt werden. Damit wird eine Tabellen-Aufteilung benutzt, die später in Abschnitt 13 noch näher erläutert wird.

Hier haben wir eine Tabelle mit String-Variablen vorgestellt, sodaß die einzigen Zahlen die Indizes in jedem Variablennamen darstellen. Nun können jedoch sowohl String- als auch numerische Tabellen benutzt werden. In den folgenden Beispielen werden numerische Tabellen vorgestellt.

Programm 1

Wir wollen hier simulieren, wie ein Würfelpaar 400mal geworfen wird. Schreiben Sie auf, wie oft jede mögliche Augensumme von 2 bis 12 vorkommt.

```
100 REMark Würfelspiel1
110 LET zwei = 0:drei = 0:vier = 0:fünf = 0:sechs= 0
120 LET sieben = 0:acht = 0:neun = 0:zehn = 0:elf = 0:
        zwölf = 0
130 FOR werfen = 1 TO 400
140   LET wurf1 = RND(1 TO 6)
```

```
150   LET wurf2 = RND(1 TO 6)
160   LET augenzahl = wurf1 + wurf2
170   IF augenzahl = 2 THEN LET zwei = zwei + 1
180   IF augenzahl = 3 THEN LET drei = drei + 1
190   IF augenzahl = 4 THEN LET vier = vier + 1
200   IF augenzahl = 5 THEN LET fünf = fünf + 1
210   IF augenzahl = 6 THEN LET sechs = sechs + 1
220   IF augenzahl = 7 THEN LET sieben = sieben + 1
230   IF augenzahl = 8 THEN LET acht = acht + 1
240   IF augenzahl = 9 THEN LET neun = neun + 1
250   IF augenzahl = 10 THEN LET zehn = zehn + 1
260   IF augenzahl = 11 THEN LET elf = elf + 1
270   IF augenzahl = 12 THEN LET zwölf = zwölf + 1
280 END FOR werfen
290 PRINT ! zwei ! drei ! vier ! fünf ! sechs
300 PRINT ! sieben ! acht ! neun ! zehn ! elf ! zwölf
```

In dem obigen Programm haben wir elf einfache Variablen benutzt, in denen jeweils die Anzahl von Würfen gespeichert wird. Werden die am Ende ausgedruckten Zahlen gezeichnet, so werden Sie ein etwa dreieckiges Balkendiagramm erhalten. Die höheren Anzahlen sind für die Augenzahlen 6, 7, 8 und die niedrigen Anzahlen für die Augenzahlen 2 und 12. Wie jeder Würfelspieler weiß, gibt dies die Häufigkeit des mittleren Augenzahlenbereichs (6, 7, 8) und die Seltenheit der Augenzahlen 2 und 12 wieder.

Programm 2

```
100 REMark Würfelspiel2
110 DIM anzahl(12)
120 FOR werfen = 1 TO 400
130    LET wurf_1 = RND(1 TO 6)
140    LET wurf_2 = RND(1 TO 6)
150    LET augenzahl = wurf_1 + wurf_2
160    LET anzahl(augenzahl) = anzahl(augenzahl) + 1
170 END FOR werfen
180 FOR nummer = 2 to 12 : PRINT anzahl(nummer)
```

In der ersten **FOR**-Schleife, in der *werfen* benutzt wird, ist der Index der Tabellen-Variablen *augenzahl*. Dies bedeutet, daß der richtige Tabellenindex automatisch für eine Erhöhung der Anzahl nach jedemWurf ausgewählt wird. Stellen Sie sich nun die Tabelle *anzahl* als Gruppe von Ablagefächern mit den Nummern 2 bis 12 vor. Wann immer eine bestimmte Augenzahl auftritt, wird die Anzahl dieser Augenzahl erhöht, indem ein Stein in das entsprechende Ablagefach geworfen wird.

In der zweiten (kürzeren) **FOR**-Schleife ist der Index gleich *augenzahl*. Während sich der Wert von *nummer* von 2 bis 12 ändert, werden sämtliche Werte der Anzahlen ausgedruckt.

Hier wird darauf hingewiesen, daß in der **DIM**-Anweisung für eine numerische Tabelle nur die Anzahl erforderlicher Variablen deklariert zu werden braucht. Hier stellt sich die Frage nach der Maximallänge, wie bei einer String-Tabelle, nicht.

Haben Sie schon mit anderen BASIC-Versionen gearbeitet, so werden Sie sich fragen, was mit der **NEXT**-Anweisung geschehen ist. Sämtliche SuperBASIC-Strukturen enden mit **END** plus einer bestimmten Angabe. Dies ist konsistent und auch sinnvoll. Die **NEXT**-Anweisung spielt jedoch auch eine Rolle, wie Sie in den späteren Abschnitten noch sehen werden.

TEST ZU KAPITEL 6

Bei diesem Test können Sie maximal 16 Punkte erzielen. Vergleichen Sie Ihr Testergebnis mit den Antworten auf Seite 111.

1. Legen Sie zwei Schwierigigkeiten dar, die anstehen, wenn die für ein Programm benötigten Daten zu zahlreich werden, und Sie versuchen, diese ohne Tabellen zu verarbeiten. (Zwei Punkte).
2. In einer Tabelle haben z. B. zehn Variablen denselben Namen. Wie können Sie dann die Variablen unterscheiden?
3. Was müssen Sie normalerweise in einem Programm tun, bevor Sie eine Tabellenvariable benutzen können?

4. Wie lautet das andere Wort für die Nummer, mit der eine bestimmte Variable einer Tabelle von den anderen Variablen mit demselben Namen unterschieden werden kann?
5. Können Sie sich zwei Situationen im täglichen Leben vorstellen, die dem Konzept der Tabelle bei der Programmierung entsprechen? (Zwei Punkte)
6. Eine **REPeat**-Schleife wird beendet, sobald eine bestimmte Bedingung zur Ausführung einer **EXIT**-Anweisung führt. Wodurch wird eine **FOR**-Schleife beendet?
7. Eine **REPeat**-Schleife benötigt einen Namen, so daß sie mit **EXIT** ordnungsgemäß beendet werden kann. Eine **FOR**-Schleife verfügt ebenfalls über einen Namen. Welche andere Funktion hat jedoch der Name einer **FOR**-Schleife noch?
8. Wie lauten die beiden Begriffe, mit denen die Variable beschrieben wird, die gleichzeitig den Namen einer **FOR**-Schleife darstellt? (Zwei Punkte)
9. Die Werte einer Schleifenvariablen ändern sich automatisch, während eine **FOR**-Schleife ausgeführt wird. Nennen Sie eine wichtige Benutzungsmöglichkeit dieser Werte.
10. Welche der folgenden Angaben haben die lange Form der **REPeat**-Schleifen und die lange Form der **FOR**-Schleifen gemeinsam? Für jedes der vier folgenden Elemente sagen Sie entweder, ob beide Schleifen dieses gemeinsam haben, oder welche Art von Schleife dieses Element aufweist.

    a) Ein öffnender Befehl bzw. eine öffnende Anweisung.
    b) Ein schließender Befehl bzw. eine schließende Anweisung.
    c) Ein Schleifen-Name.
    d) Eine Schleifenvariable oder Steuervariable.
    (Vier Punkte)

AUFGABEN ZU KAPITEL 6

1. Benutzen Sie eine **FOR**-Schleife, um eine von vier Zahlen 1, 2, 3, 4 zufällig in fünf Tabellen-Variablen zu setzen:

   karte(1), karte(2), karte(3), karte(4), Karte(5)

   Es spielt keine Rolle, wenn eine der vier Zahlen wiederholt wird. Benutzen Sie eine zweite **FOR**-Schleife, um die Werte der fünf Kartenvariablen auszugeben.
2. Stellen Sie sich vor, daß die vier Zahlen 1, 2, 3, 4 'Herz', 'Kreuz', 'Karo', 'Pik' darstellen. Welche zusätzlichen Programmzeilen müssen eingefügt werden, um eine Ausgabe in Form dieser Worte und nicht in Zahlen zu erhalten?
3. Mit einer **FOR**-Schleife setzen Sie fünf Zufallszahlen im Bereich von 1 bis 13 in eine Tabelle mit fünf Variablen:

   karte(1), karte(2), karte(3), karte(4) und *karte(5)*

   Mit einer zweiten **FOR**-Schleife geben Sie die Werte der fünf Kartenvariablen aus.
4. Stellen Sie sich vor, daß die in Aufgabe 1 erzeugten Zufallszahlen Karten darstellen. Schreiben Sie die zusätzlichen Anweisungen auf, die zu folgender Ausgabe führen würden:

Zahl	Ausgabe
1	Das Wort "As"
2 bis 10	Die Kartenzahl
11	Das Wort "Bube"
12	Das Wort "Dame"
13	Das Wort "König"

KAPITEL 7
EINFACHE PROZEDUREN

Müssen Sie Computerprogramme zur Lösung komplexer Probleme schreiben, so haben Sie unter Umständen Schwierigkeiten, die Übersicht zu bewahren. Bei einer methodischen Problemlösung wird demzufolge ein umfangreicher oder komplexer Job in kleinere **Aufgaben** unterteilt. Diese Aufgaben werden dann solange wieder in kleinere Aufgaben unterteilt, bis die einzelnen Aufgaben so begrenzt sind, daß sie problemlos bearbeitet werden können.

Dies kann mit der Lösung schwieriger Aufgaben im menschlichen Bereich verglichen werden. Eine erfolgreiche Regierung muß Verantwortung delegieren. Der Ministerpräsident teilt die Arbeit unter seinen Ministern auf, die sie wieder in ihrem Verwaltungsapparat aufteilen,bis die Arbeiten von Einzelpersonen ohne weitere Unterteilung ausgeführt werden können. Hier gibt es nun schwierige Situationen, wie gemeinsame Dienstleistungen und Zusammenspiel auf verschiedenen Ebenen. Die hierarchische Struktur ist jedoch dominierend.

Ein guter Programmierer arbeitet ebenfalls auf diese Art und Weise. Dabei ist eine moderne Sprache, wie SuperBASIC, die eindeutig benannte Prozeduren zuläßt, wesentlich hilfreicher als ältere BASIC-Versionen, die nicht über derartige Möglichkeiten verfügen.

Eine Prozedur ist ein separat benannter Programmteil, der für eine bestimmte Aufgabe geschrieben wird. Es spielt keine Rolle, an welcher Stelle die Prozedur in dem Programm steht. Der Name der Prozedur kann wie ein Befehl benutzt werden. Dann passiert folgendes:

- die Anweisungen der Prozedur werden ausgeführt
- anschließend wird die Anweisung des Programms ausgeführt, die demProzeduraufruf folgt.

Nehmen Sie an, Sie haben eine Prozedur mit dem Namen *Quadrat*, die ein Quadrat zeichnet. Ein Programm, das diese Prozedur benutzt, arbeitet nach folgendem Schema:

Prozedurdefinition **Prozeduraufruf**

```
DEFine PROCedur Quadrat
REMark Anweisungen zum
  Zeichnen des Quadrates          <------------ Quadrat
END DEFine
        |
Zeichnet ein Quadrat
```

In der Praxis können die einzelnen Aufgaben innerhalb eines Komplexes benannt werden, bevor die einzelnen Anweisungen geschrieben werden. Beim Aufruf der Prozedur wird nur der Name benötigt, so daß das Programm als Ganzes geschrieben werden kann, bevor alle Teilaufgaben programmiert sind.

Bei einer anderen Methode werden die einzelnen Aufgaben zuerst programmiert und getestet. Werden sie einwandfrei ausgeführt, so können Sie die Einzelheiten vergessen und brauchen sich nur an den Namen und die Funktion der Prozedur zu erinnern.

Beispiel

Das folgende Beispiel könnte problemlos ohne Prozeduren geschrieben werden. Es zeigt jedoch in einem relativ einfachen Kontext, wie Prozeduren benutzt werden können. Nahezu jede Aufgabe kann auf ähnliche Art und Weise unterteilt werden. Dies bedeutet, daß Sie sich niemals mit mehr als sagen wir 5 bis 30 Zeilen gleichzeitig plagen müssen. Können Sie 30 Zeilen umfassende Programme einwandfrei schreiben und Prozeduren verarbeiten, so können Sie auch problemlos Programme mit 300 Zeilen schreiben.

Hier können Sie Politikern oder anderen Personen, die den Eindruck technischen Wissens vermitteln möchten, ohne in Wirklichkeit irgend etwas davon zu verstehen, beim Phrasendreschen helfen. Speichern Sie die folgenden Wörter in drei Tabellen und erzeugen Sie dann zehn zufällige fertige Phrasen.

adjek1$	adjek2$	subst$
vollständige	modernste	Systme
systematische	wissensbasierte	Maschinen
intelligente	kompatible	Computer
kontrollierte	kybernetische	Rechenanlagen
automatisierte	benutzerfreundliche	Kapazitäten
synchronisierte	parallele	Mikro-Chips
functionale	lernende	Konsole
optionale	anpassungsfähige	Programmierer
positive	modulare	Pakete
ausgeglichene	strukturierte	Datenzentren
integrierte	logik-orientierte	Datenverarbeiter
koordinierte	kartei-orientierte	Prozessoren
wissenschaftliche	normierte	Rückkoppelungen

ANALYSE

Nun werden wir ein Programm schreiben, mit dem zahn vorgefertigte Sätze erzeugt werden. Die einzelnen Programmstationen sind:

1. Speichern Sie die Wörter in drei String-Tabellen.
2. Wählen Sie drei Zufallszahlen, die als Indizes der Tabellen-Variablen benutzt werden.
3. Drucken Sie die Phrase aus.
4. Wiederholen Sie die Schritte 2 und 3 zehn Mal.

FESTLEGEN VON VARIABLEN

Wir benennen drei Tabellen, von denen die beiden ersten Adjektive oder als Adjektiv benutzte Wörter enthalten. Die dritte Tabelle enthält die Hauptwörter. In jedem Abschnitt gibt es 13 Wörter, wobei das längste Wort 16 Zeichen umfaßt.

Tabelle	Zweck
adjek1$(13,17)	Erste Adjektive
adjek2$(13,19)	Zweite Adjektive
subst$(13,17)	Hauptwörter

PROZEDUREN

Hier werden drei Prozeduren benutzt, die den drei angegebenen Aufgaben entsprechen.

Mit *daten_speichern* werden die drei Gruppen mit 13 Wörtern gespeichert. Mit *wort_wahl* werden drei Zufallszahlen im Bereich von 1 bis 13 gewählt.

Mit *zusammensetzung* wird eine Phrase ausgedruckt.

HAUPTPROGRAMM

Das Hauptprogramm ist sehr einfach, da die Hauptarbeit von den Prozeduren übernommen wird.

Deklaration der Tabellen **(DIM)**
Daten_speichern
FOR zehn Begriffe
Wort_wahl
Zusammensetzung
END

Programm

```
100 REMark ********************
110 REMark * PHRASENDRESCHEREI*
120 REMark ********************
130 DIM adjec1$(13,17), adjec2$(13,19),subst$(13,17)
140 daten_speichern
150 FOR phrase = 1 TO 10
160   wort_wahl
170   zusammensetzung
180 END FOR phrase
190 REMark ***************************
200 REMark *Definition der Prozeduren
210 REMark ***************************
220 DEFine PROCedure daten_speichern
230   REMark *** Procedur um DATA zu speichern ***
```

```
240 RESTORE 420
250 FOR word = 1 TO 13
260 READ adjec1$(wort), adjec2$(wort),subst$(wort)
270 END FOR wort
280 END DEFine
290 DEFine PROCedure wort_wahl
300 REMark *** Prozedur um Wörter zu wählen ***
310 LET ad1 = RND(1 TO 13)
320 LET ad2 = RND(1 TO 13)
330 LET S = RND(1 TO 13)
340 END DEFine
350 DEFine PROCedure zusammensetzung
360 REMark *** Procedur um Phrase auszudrucken ***
370 PRINT ! adjec1$(ad1) ! adjec2$(ad2) ! subst$(S)
380 END DEFine
390 REMark ****************
400 REMark * Program DATA *
410 REMark ****************
420 DATA ''vollständige'',''modernste'',''Systeme''
430 DATA ''systematische'',''wissensbasierte'',''Maschine''
440 DATA ''intelligente'',''kompatible'',''Computer''
450 DATA ''kontrollierte'',''kybernetische'',''Rechenan-
lage''
460 DATA ''automatisierte'',''anwenderfreundli-
che'',''Kapazitäten''
470 DATA ''synchronisierte'',''parallele'',''Mikro-
Chips''
480 DATA ''funktionale'',''lernende'',''Konsole''
490 DATA ''optionale'',''anpassungsfähige'',''Pro-
grammierer''
500 DATA ''positive'',''modulare'',''Pakete''
510 DATA ''ausgeglichene'',''strukturierte'',''Da-
tenzentren''
520 DATA ''integrierte'',''logikorientierte'',''Da-
tenverarbeiter''
530 DATA ''koordinierte'',''karteiorientierte-
'',''Prozessoren''
540 DATA ''wissenschaftliche'',''normierte'',''Rück-
koppelungen''
```

```
funktionale logikorientierte Rückkoppelungen
vollständige parallele Prozessoren
optionale normierte Kapazitäten
positive anwenderfreundliche Mikro-Chips
positive lernende Datenverarbeiter
synchronisierte karteiorientierte Datenzentren
kontrollierte modernste Rückkoppelungen
kontrollierte karteiorientierte Pakete
```

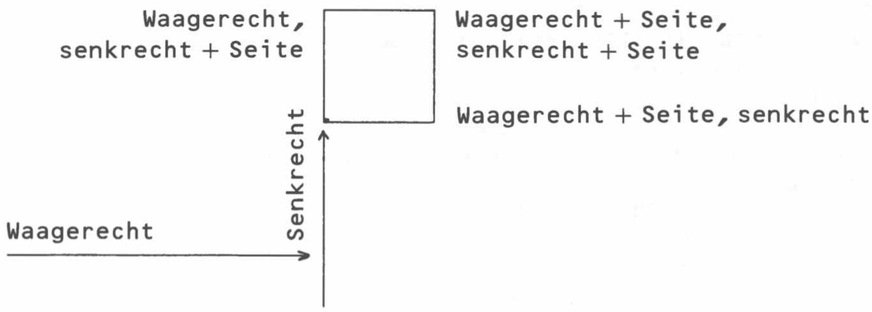
ÜBERGABE VON INFORMATION AN PROZEDUREN

Angenommen, es sollen Quadrate mit verschiedener Größe und verschiedenen Farben in verschiedenen Positionen auf dem Bildschirm gezeichnet werden.

Hierzu definieren wir eine Prozedur, Quadrat, für die vier Angaben erforderlich sind:

- Länge einer Seite
- Farbe (Farbcode)
- Position (waagerecht und senkrecht).

Die Position der Quadrate wird durch Angabe von zwei Werten, waagerecht und senkrecht, festgelegt. Mit ihnen wird die untere linke Ecke des Quadrates wie nachfolgend dargestellt festgelegt.



Die Farbe des Quadrates kann problemlos festgelegt werden. Bei dem Quadrat selbst werden jedoch die Werte von *Seite*, *Waagerecht* und *Senkrecht* wie folgt benutzt.

```
200 DEFine PROCedure quadrat(seite,waag,senk)
210   LINE waag,senk TO waag+seite,senk
220   LINE TO waag+seite,senk+seite
230   LINE TO waag,senk+seite TO waag,senk
240 END DEFine
```

Damit diese Prozedur ausgeführt werden kann, müssen Werte für *seite, waag* und *senk* angegeben werden. Diese Werte werden angegeben, wenn die Prozedur aufgerufen wird. So könnten Sie beispielsweise das folgende Hauptprogramm hinzufügen, um ein grünes Quadrat mit der Seitenlänge 20 zu erhalten.

```
100 PAPER 7: CLS
110 INK 4
120 quadrat 20,50,50
```

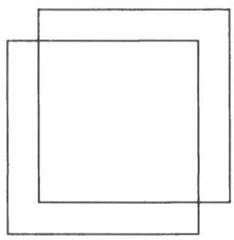
Die Zahlen 20,50,50 werden als Parameter bezeichnet. Sie werden an die, in der Prozedurdefinition benannten Variablen übergeben:

Quadrat 20,50,50

DEFine PROCedure Quadrat(seite,waag,senk)

Die Zahlen 20,50,50 werden als aktuelle Parameter bezeichnet. In diesem Fall handelt es sich um Zahlen, es könnte sich jedoch auch um Variablen oder Ausdrücke handeln. Die Variablen *seite, waag, senk* werden als formale Parameter bezeichnet. Hier muß es sich um Variablen handeln, da sie Werte "empfangen".

Ein interessanteres Hauptprogramm benutzt dieselbe Prozedur, um ein zufälliges Muster mit farbigen Quadratpaaren zu erzeugen. Jedes Quadratpaar wird erhalten, indem das zweite Quadrat um ein Fünftel der Seitenlänge des ersten waagerecht und senkrecht verschoben wird:



Angenommen, die Prozedur **Quadrat** steht noch in Zeile 200, so hat das folgende Programm den gewünschten Effekt.

```
100 REMark Quadratmuster
110 PAPER 7 : CLS
120 FOR paare=1 TO 20
130   INK RND(5)
140   LET seite=RND(10 TO 20)
150   LET waag=RND(50) : senk = RND(70)
160   quadrat seite,waag,senk
170   LET waag=waag+seite/5 : senk=senk+seite/5
190 END FOR paare
```

Prozeduren haben folgende Vorteile:

1. Sie können denselben Code mehr als ein Mal in demselben Programm oder in anderen Programmen benutzen.
2. Sie können eine Aufgabe in Teilaufgaben aufteilen und für jede Teilaufgabe Prozeduren schreiben. Dadurch werden Analysen und Programmaufbau vereinfacht.
3. Prozeduren können separat getestet werden. Dadurch kann das Austesten vereinfacht werden.
4. Durch bedeutungsvolle Prozedurnamen kann ein Programm leserlich gestaltet werden.

Sobald Sie sich einmal an geeignet benannte Prozeduren und den Umgang mit Parametern gewöhnt haben, werden Sie feststellen, daß Ihre Fähigkeiten zur Problemlösung und Programmierung wesentlich verbessert wurden.

TEST ZU KAPITEL 7

Bei diesem Test können Sie maximal 15 Punkte erzielen. Vergleichen Sie Ihr Testergebnis mit den Antworten auf Seite 112.

1. Wie lösen wir im täglichen Leben das Problem umfangreicher und komplexer Aufgaben?
2. Wie kann dieses Prinzip auf die Programmierung angewandt werden?
3. Welches sind die beiden wichtigsten Funktionen einer einfachen Prozedurdefinition? (Zwei Punkte)
4. Welche beiden Hauptauswirkungen hat die Benutzung eines Prozedurnamens für en 'Aufruf' der Prozedur? (Zwei Punkte)
5. Welchen Vorteil hat die Benutzung von Prozedurnamen in einem Hauptprogramm, bevor die Prozedurdefinitionen geschrieben werden?
6. Welchen Vorteil hat das Schreiben einer Prozedurdefinition vor der Benutzung ihres Namens in einem Hauptprogramm?
7. Wie kann die Benutzung von Prozeduren einem Programmierer der 30zeilige Programme schreiben kann dabei helfen, wesentlich größere Programme zu schreiben?
8. Einige Programme benutzen mehr Speicherplatz bei der Definition von Prozeduren. Unter welchen Umständen sparen Prozeduren jedoch Speicherplatz?
9. Nennen Sie zwei Möglichkeiten, mit denen Angaben von einem Hauptprogramm in eine Prozedur übergeben werden können. (Zwei Punkte)
10. Was ist ein aktueller Parameter?
11. Was ist ein formaler Parameter?

AUFGABEN ZU KAPITEL 7

1. Schreiben Sie eine Prozedur, mit der eine der vier folgenden Kartenfarben ausgegeben wird: 'Herz', 'Kreuz', 'Karo' oder 'Pik'. Rufen Sie die Prozedur fünf Mal auf, um fünf zufällige Farben zu erhalten.
2. Schreiben Sie ein weiteres Programm für Aufgabe 1, wobei Sie eine Zahl im Bereich von 1 bis 4 als Parameter zur Festlegung des Ausgabewortes benutzen. Haben Sie dies schon getan, so versuchen Sie, das Programm ohne Parameter zu schreiben.
3. Schreiben Sie eine Prozedur, mit der der Wert einer Karte, d.h. eine Zahl im Bereich von 2 bis 10, oder eines der Worte "As", "Bube", "Dame", "König" ausgegeben wird.
4. Schreiben Sie ein Programm, das diese Prozedur fünf Mal aufruft, so daß fünf zufällige Werte ausgegeben werden.
5. Schreiben Sie das Programm in Aufgabe 3 neu, wobei Sie eine Zahl im Bereich von 1 bis 13 als Parameter benutzen, der an die Prozedur übergeben wird. Haben Sie diese Methode schon beim ersten Mal benutzt, so versuchen Sie, das Programm ohne Parameter zu schreiben.
6. Schreiben Sie das eleganteste Programm mit Prozeduren, um an vier Spieler jeweils fünf Karten auszugeben. Machen Sie sich keine Gedanken, um doppelte Karten. Elegant steht hier für die entsprechende Mischung von Lesbarkeit, Kürze und Effizienz. Andere Benutzer oder andere Umstände messen diesen drei Eigenschaften eine andere Bedeutung zu. Gelegentlich arbeiten diese Eigenschaften auch gegeneinander.

KAPITEL 8

VON BASIC ZU SUPERBASIC

Sind Sie schon mit einer der früheren BASIC-Versionen vertraut, so können Sie unter Umständen die ersten sieben Kapitel dieses Abschnitts überspringen und erst diesen Abschnitt als Verbindung zwischen Ihren schon vorhandenen Kenntnissen und den restlichen Abschnitten lesen. Sollten Sie in diesem Fall dennoch auf Schwierigkeiten stoßen, so können Sie jederzeit in den vorhergehenden Abschnitten nachschlagen.

Haben Sie die vorhergehenden Abschnitte durchgearbeitet, so werden Sie mit diesem Abschnitt keine Schwierigkeiten haben. Neben der Vorstellung einiger neuer Ideen, werden Sie feststellen, daß dieser Abschnitt einen interessanten Überblick über die Entwicklung von BASIC bietet. Neben den Einrichtungen zur Programmstrukturierung erweitert SuperBASIC auch die Grenzen der Bildschirmdarstellung, Bearbeitung, Ausführung und Grafik. Kurz gesagt ist SuperBASIC eine Kombination aus Benutzerfreundlichkeit und Rechenleistung, die bislang noch nicht dagewesen ist.

Gehen Sie also von BASIC zu SuperBASIC, so gehen Sie nicht nur zu einer leistungsfähigeren, hilfreichern Sprache, sondern auch zu einer bemerkenswert erweiterten Computer-Leistung.

Nun werden wir einige der Hauptfunktionen von SuperBASIC und einige der Funktionen erläutern, die SuperBASIC von den anderen BASIC-Versionen unterscheiden.

ALPHABETISCHE VERGLEICHE

Die üblichen einfachen alphabetischen Vergleiche sind möglich. So können Sie:

```
LET tier1$ = ''Hund''
LET tier2$ = ''Katze''
IF tier1$ < tier2$ THEN PRINT ''MIAU''
```

schreiben. Auf dem Bildschirm wird MIAU ausgegeben, da das Symbol < in diesem Zusammenhang folgende Bedeutung hat:

früher (näher bei A im Alphabet)

SuperBASIC nimmt die Vergleiche auf sensible Art und Weise vor. So erwarten Sie beispielsweise, daß:

'hund' vor 'KATZE'

und

'ERD98L' vor 'ERD746L'

steht.

Bei einer einfachen Lösung, bei der blind die interne Zeichencodierung benutzt wird, würde in den beiden obigen Fällen das "falsche" Ergebnis ausgegeben. Versuchen Sie nun jedoch das folgende Programm, mit dem die "frühere"der beiden Zeichenfolgen ermittelt wird.

```
100 INPUT string1$, string2$
110 IF string1$ < string2$ THEN PRINT string1$
120 IF string1$ = string2$ THEN PRINT ''Gleiche Strings''
130 IF string1$ > tier 2$ THEN PRINT string2$
```

EINGABE		AUSGABE
hund	katze	hund
hund	KATZE	hund
ERD98L	ERD746L	ERD98L
ABC	abc	ABC

In dem Abschnitt *Begriffe* wird genau beschrieben, wie Vergleiche von Strings bei SuperBASIC ausgeführt werden.

VARIABLE UND NAMEN

Die meisten BASIC-Versionen verfügen über numerische und String-Variablen. Wie bei anderen BASIC-Versionen wird ein String-Variablenname bei SuperBASIC durch ein Dollarzeichen am Ende gekennzeichnet. Beispiele:

Numerische Variable:	zahl	String:	*wort$*
	summe		*hoch_str$*
	gesamt		*woche_tag$*

Möglicherweise haben Sie so bedeutungsvolle Variablennamen noch nicht angetroffen, obwohl einige der jüngsten BASIC-Versionen sie zulassen. Die Regeln für Namen in SuperBASIC stehen in dem Kapitel *Begriffe*. Ein Name hat eine Länge von maximal 255 Zeichen. Die Auswahl der Namen ist eine persönliche Entscheidung. Gelegentlich vermitteln die längeren Namen dem Leser eher die Funktion eines Programms. Aber sie verursachen etwas mehr Arbeit beim eintippen.
Im allgemeinen ist *Spaten* jedoch bedeutungsvoller als der Ausdruck *Gerät zur Gartenbearbeitung*. Kurze Wörter werden stets bevorzugt, wenn sie dennoch die Bedeutung eindeutig wiedergeben. Sehr kurze Wörter oder einzelne Buchstaben sollten jedoch nur vereinzelt benutzt werden. Variablennamen wie *X,Z,P3,Q2* führen zu einer Abstraktionsebene, mit der die meisten Menschen nichts anfangen können.

GANZZAHLIGE VARIABLEN

Bei SuperBASIC sind **ganzzahlige** Variable zulässig, die nur Werte von ganzen Zahlen annehmen können. Diese werden mit einem Prozentzeichen am Ende gekennzeichnet:

zahl%
nummer%
nächste_seite%

Nun gibt es zwei Arten von numerischen Variablen. Die andere Art, die auch gebrochene Werte annehmen kann, wird als **Gleitkomma-Variable** bezeichnet. Sie können folgendes schreiben:

```
LET preis = 9
LET kosten = 7.31
LET zahl% = 13
```

Schreiben Sie jedoch:

```
LET zahl% = 5.43
```

so wird der Wert von *zahl%$* zu 5. Andererseits führt der Befehl:

```
LET zahl% = 5.73
```

zu einem Wert von 6 für *zahl%*. Wie Sie sehen, tut SuperBASIC sein bestes, um zu der nächsten ganzen Zahl auf- bzw. abzurunden.

DATENTYP-UMWANDLUNG

Das Prinzip, dem Benutzer stets intelligent unter die Arme zu greifen anstatt eine Fehlermeldung oder ein unerwünschtes Ergebnis auszugeben, geht noch weiter. Hat eine String-Variable *mark$* beispielsweise den Wert

```
'64'
```

so erzeugt der Befehl:

```
LET ergebnis = note$
```

einen numerischen Wert von 64 für Augenzahl. Andere BASIC-Versionen würden hier wahrscheinlich stoppen und eine Fehlermeldung wie:

'Type mis-match'
oder 'Nonsense in BASIC'

ausgeben. Kann der String nicht umgewandelt werden, so wird eine Fehlermeldung ausgegeben.

LOGISCHE VARIABLEN UND EINFACHE PROZEDUREN

Nun gibt es bei SuperBASIC noch einen anderen Variablentyp oder zumindest scheint es so. Betrachten Sie die SuperBASIC -Anweisung:

```
IF windig THEN drachen_steigen
```

In anderen BASIC-Versionen würden Sie wahrscheinlich:

```
IF w=1 THEN GOSUB 300
```

schreiben. In diesem Fall ist *w = 1* eine Bedingung oder ein logischer Ausdruck, der entweder wahr oder falsch ist. Ist er wahr, so würde ein Unterprogramm ausgeführt, das ab Zeile 300 beginnt. Dieses Unterprogramm kann sich wohl mit dem Drachenfliegen beschäftigen, anhand der obigen Zeile können Sie dies jedoch nicht feststellen. Ein bedachter Programmierer würde:

```
IF w=1 THEN GOSUB 300 : REM drachen_steigen
```

schreiben, um das Programm einfacher lesen zu können. Die SuperBASIC-Anweisung kann jedoch so wie sie angegeben wird, problemlos gelesen werden. Der Name *windig* wird als wahr oder falsch interpretiert, obwohl es sich in Wirklichkeit um eine Gleitkommavariable handelt. Ein Wert von 1 oder jeder von Null verschiedene Wert wird als *wahr* interpretiert. Null wird als falsch ausgelegt. So hat das einzelne Wort, *windig*, dieselbe Auswirkung wie eine Bedingung eines logischen Ausdrucks.

Das andere Wort, *drachen_steigen* ist eine Prozedur. Sie führt dieselbe Aufgabe wie **GOSUB 300** aus, jedoch wesentlich besser.

Das folgende Programm soll die Vorstellung von logischen Variablen und der einfachsten Art einer Prozedur vermitteln.

```
100 INPUT windig
110 IF windig THEN drachen_steigen
120 IF NOT windig THEN sachen_ordnen
130 DEFine PROCedure drachen_steigen
140   PRINT ''Sieh wie er in der Luft tanzt''
150 END DEFine
160 DEFine PROCedure sachen_ordnen
170   PRINT ''Sortier den Müll aus''
180 END EFine
```

EINGABE	AUSGABE
0	Sortier den Müll aus
1	Sieh wie er in der Luft tanzt
2	Sie wie er in der Luft tanzt
−2	Sieh wie er in der Luft tanzt

Aus diesem Beispiel können Sie ersehen, daß nur Null die Bedeutung falsch hat. Normalerweise würden Sie keine Prozeduren mit nur einer ausführbaren Anweisung schreiben. Anhand, des Programms werden jedoch die Vorstellung und die Syntax in einem sehr einfachen Kontext dargestellt. Mehr über die Prozeduren später in diesem Abschnitt.

LET-ANWEISUNGEN

In SuperBASIC muß **LET** bei einer Zuweisung nicht angegeben werden. Es wird jedoch in diesem Handbuch benutzt, um die beiden Benutzungsmöglichkeiten von = darzulegen. Bei:

```
LET zahl = 3
```

und

```
IF zahl = 3 THEN EXIT
```

hat das = eine unterschiedliche Bedeutung. Mit **LET** kann dies verdeutlicht werden. Gibt es jedoch mehrere **LET**-Anweisungen, die eine einfache Aufgabe wie das Setzen von Ausgangswerten ausführen, so kann eine Ausnahme gemacht werden.

Zum Beispiel:

```
100 LET erste = 0
110 LET zweite = 0
120 LET dritte = 0
```

kann folgendermaßen neu geschrieben werden:

```
100 LET erste = 0 : zweite = 0 : dritte = 0
```

ohne daß Deutlichkeit oder Stil beeinträchtigt werden.

Der Doppelpunkt ":" dient als Trennzeichen zwischen Anweisungen, die in derselben Zeile stehen.

DER BILDSCHIRM

In einem der folgenden Abschnitte werden wir erläutern, wie andere Grafik-Funktionen, wie beispielsweise das Zeichnen von Kreisen, gehandhabt werden können. Hier werden wir uns jedoch mit den pixel-orientierten Anweisungen befassen. Mit einem der folgenden Befehle kann zwischen zwei Betriebsarten gewählt werden:

Niedrige Auflösung 8-Farb-Modus 256 x 256 Pixel	MODE 256 MODE 8

Hohe Auflösung 4-Farb-Modus 512 x 256 Pixel	MODE 512 MODE 4

In beiden Betriebsarten werden die Pixel innerhalb des folgenden Bereichs adressiert:

0–511 waagerecht
und 0–255 senkrecht

Da bei **MODE 8** waagerecht nur die Hälfte der Pixel wie in **MODE 4** vorhanden ist, sind die Pixel in **MODE 8** doppelt so breit wie in **MODE 4**. Deshalb kann in **MODE 8** jedes Pixel mit zwei Koordinaten angegeben werden. Zum Beispiel:

0 oder 1   2 oder 3   510 oder 511.

Dies bedeutet auch, daß Sie denselben Zahlenbereich für die Adressierung von Pixeln unabhängig von der Betriebsart benutzen können. Benutzen Sie stets 0 bis 511 für die waagerechte und 0 bis 255 für die senkrechte Richtung.

Benutzen Sie ein Fernsehgerät, so können nicht alle Pixel angezeigt werden.

FARBEN

Folgende Farben stehen zur Verfügung:

MODUS 256	Code	MODUS 512
Schwarz	0	Schwarz
Blau	1	
Rot	2	Rot
Magenta	3	
Grün	4	Grün
Cyan	5	
Gelb	6	Weiß
Weiß	7	

Magenta ist ein Himbeerrot, Cyan ist ungefähr Türkisfarben.

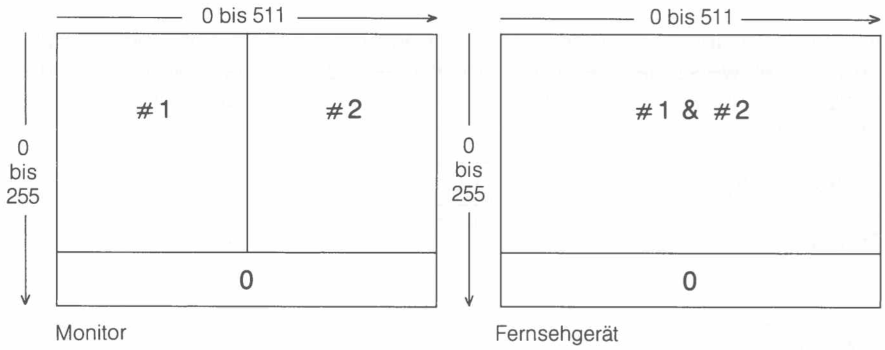
In der *hochauflösenden Betriebsart* kann jede Farbe mit einem von zwei Codes ausgewählt werden. Sie werden später sehen, wie ein faszinierender Farbbereich und faszinierende Punktmuster erzeugt werden können, wenn Sie einen hochwertigen Farbmonitor haben.

Einige der Befehle für die Bildschirmdarstellung lauten:

INK *Farbe*	Vordergrundfarbe
BORDER *Breite, Farbe*	Zeichnet eine Umrandung um den Bildschirm oder das Fenster
PAPER *Farbe*	Hintergrundfarbe
BLOCK *Breite, Höhe, waagerecht, senkrecht, Farbe*	Zeichnet ein farbiges Rechteck, dessen obere linke Ecke sich in der Position waagerecht, senkrecht befindet.

BILDSCHIRM-ORGANISATION

Schalten Sie den QL ein, so ist die Bildschirmanzeige wie nachfolgend dargestellt in drei Bereiche unterteilt, die als *Fenster* bezeichnet werden. Um diese Fenster in den Bereich zu setzen, der von einem Fernsehbildschirm abgedeckt wird, können einige Pixel in der Umrandung im Fernseh-Modus nicht benutzt werden.



Die Fenster werden durch #0, #1 und #2 gekennzeichnet, so daß Sie verschiedene Effekte verschiedenen Fenstern zuweisen können. Mit:

```
CLS
```

wird Fenster #1 gelöscht (dies wird vom System ausgewählt). Soll also der linke Bereich gelöscht werden, so müssen Sie:

```
CLS #2
```

eingeben. Wird eine andere Papier-Farbe (Hintergrundfarbe) gewünscht, so geben Sie folgenden Befehl ein, wenn Sie die Farbe Grün wünschen:

```
PAPER 4: CLS
```

Soll Fenster #2 auf die Hintergrundfarbe Grün gelöscht werden, so geben Sie folgenden Befehl ein:

```
PAPER 4: CLS
```

or

```
PAPER #2,4: CLS #2
```

Die Zahlen #0, #1, #2 werden als *Kanal-Nummern* bezeichnet. In diesem Fall helfen Sie Ihnen dabei, bestimmte Ausgaben in das Fenster Ihrer Wahl zu leiten. Später werden Sie noch feststellen, daß die Kanalnummern noch viele andere Funktionen haben. Für den Augenblick denken Sie jedoch nur daran, daß alle folgenden Anweisungen eine Kanalnummer aufweisen können. In der dritten Spalte wird der Standardkanal angegeben – d. h. der Kanal, den das System auswählt, wenn Sie keinen Kanal angeben.
Hier wird darauf hingewiesen, daß sich die Fenster überlagern können. Bei einem Fernsehbildschirm überlagert das System automatisch die Fenster #1 und #2, so daß mehr Zeichenpositionen pro Zeile für die Programmauflistungen zur Verfügung stehen.

Befehl	Funktion	Standard-Fenster
AT	Zeichenposition	#1
BLOCK	Zeichnet einen Block	#1
BORDER	Zeichnet eine Umrandung	#1
CLS	Löscht den Bildschirm	#1
CSIZE	Zeichengröße	#1
CURSOR	Positioniert den Cursor	#1
FLASH	Setzt/Löscht das Blinken	#1
INK	Schriftfarbe	#1
OVER	Drucken und Grafiken	#1
PAN	Verschiebt den Bildschirm seitlich	#1
PAPER	Hintergrundfarbe	#1
RECOL	Ändert die Farbe	#1
SCROLL	Verschiebt den Bildschirm senkrecht	#1
STRIP	Hintergrund für das Ausdrucken	#1
UNDER	Unterstreichung	#1
WINDOW	Ändert das bestehende Fenster	#1
LIST	Listet das Programm auf	#1
DIR	Listet das Inhaltsverzeichnis auf	#2
PRINT	Druckt Zeichen aus	#1
INPUT	Tastatureingabe	#1

Anweisungen oder direkte Befehle stehen in Fenster #0.

Für weitere Angaben über die Syntax oder Benutzung dieser Befehle wird auf andere Teile dieses Handbuchs verwiesen.

RECHTECKE UND LINIEN

Mit dem folgenden Programm wird ein grünes Rechteck bei MODE 256 auf rotem Papier mit einem gelben Rand in einer Breite von einem Pixel gezeichnet. Die obere linke Ecke des Rechtecks befindet sich bei den Pixel-Koordinaten 100,100 (siehe QL *Begriffe*). Das Rechteck hat eine Breite von 80 Einheiten waagrecht (40 Pixel bei MODE 256) und eine Höhe von 20 Einheiten senkrecht (20 Pixel bei MODE 256).

```
100 REMark Rechteck
110 MODE 256
120 BORDER 1,6
130 PAPER 2 : CLS
140 BLOCK 80,20,100,100,4
```

Im **MODUS 256** muß mit etwas Vorsicht vorgegangen werden,da die waagerechten Werte von 0 bis 511 gehen, obwohl nur 256 Pixel vorhanden sind. Wir können nicht sagen, daß der mit dem obigen Programm erzeugte Block eine Breite von 80 Pixel hat, deshalb wird von 80 Einheiten gesprochen.

EIN- UND AUSGABE

SuperBASIC verfügt über die üblichen **LET-**, **INPUT-**, **READ-** und **DATA**-Anweisungen für die Eingabe. Die **PRINT**-Anweisung handhabt den größten Teil der Textausgabe auf die übliche Art und Weise mit den Trennzeichen:

,	Legt die Ausgabe in Tabellenform fest
;	nur Trennzeichen – keine Auswirkung auf die Formatierung
\	Bewirkt einen Zeilenvorschub
!	Bewirkt ein Leerzeichen, jedoch nicht am Anfang einer Zeile. Paßt ein Element nicht in den Rest einer Zeile, so wird ein Zeilenvorschub vorgenommen.
TO	Stellt den Tabulator auf die angegebenen Spalten-Position.

SCHLEIFEN

Sie sind schon mit den beiden nachfolgend dargelegten Arten von Widerholungsschleifen vertraut:

(a) Simuliert sechs Würfe mit einem normalen sechsseitigen Würfel.

```
100 FOR werfen = 1 TO 6
110 PRINT RND(1 TO 6)
120 NEXT werfen
```

(b) Simuliert die Würfe eines Würfels, bis 6 geworfen wird.

```
100 augenzahl = RND(1 TO 6)
110 PRINT augenzahl
120 IF augenzahl <> 6 THEN GO TO 10
```

Diese beiden Programme können in SuperBASIC benutzt werden. Allerdings empfehlen wir statt dessen folgende Version. Sie führen genau dieselbe Aufgabe aus. Auch wenn Programm (b) etwas komplexer ist, gibt es einen guten Grund, dieses Programm zu bevorzugen.

(a)
```
100 FOR werfen = 1 TO 6
110   PRINT RND(1 TO 6)
120 END FOR werfen
```

(b)
```
100 REPeat werfen
110   augenzahl = RND(1 TO 6)
120   PRINT augenzahl
130   IF augenzahl = 6 THEN EXIT werfen
140 END REPeat werfen
```

Es ist sinnvoll, eine Struktur für eine Schleife vorzusehen, die bei einer Bedingung beendet wird (**REPeat**-Schleifen), sowie Schleifen, die durch einen Zähler gesteuert werden. Die grundlegende Struktur von **REPeat** sieht folgendermaßen aus:

REPeat *Name*
  Anweisungen
END REPeat *Name*

Die **EXIT**-Anweisung kann an eine beliebige Stelle in der Struktur gestellt werden. Auf sie muß jedoch ein Name folgen, damit SuperBASIC weiß, welche Schleife beendet werden muß. Zum Beispiel bewirkt:

EXIT werfen

daß das Programm bei der Anweisung hinter

END REPeat werfen.

fortgesetzt wird. Dies sieht vielleicht aus wie das Lösen einer einfachen Schraube mit einem Vorschlaghammer. Aber die **REPeat**-Struktur erweist ihre volle Leistungsfähigkeit an einer Vielzahl verschiedenartiger Aufgaben. Sie werden ihr noch oft begegnen.

Wenn Sie andere Sprachen kennen, so werden Sie feststellen, daß Sie die Aufgabe sowohl von **REPEAT**- als auch von **WHILE**-Strukturen übernimmt, und darüber hinaus noch wesentlich schwierigere Situationen meistert.

Bei SuperBASIC ist die **REPeat**-Schleife benannt, so daß eine eindeutige, klare Beendigung vorgenommen werden kann. Die **FOR**-Schleife endet wie alle SuperBASIC-Strukturen mit **END**. Ihr Name wird aus Gründen angegeben, die erst später deutlich werden.

Sie werden auch feststellen, wie diese Schleifen-Strukturen in einfachen oder komplexen Situationen für die Ausführung genau der gewünschten Aufgabe benutzt werden können. In diesem Stadium werden wir nur noch drei weitere Funktionen der Schleifen erwähnen. Sie werden Ihnen vertraut sein, wenn Sie schon mit BASIC gearbeitet haben.

Die Erhöhung der Steuervariablen bei einer **FOR**-Schleife beträgt normalerweise 1. Sie können ihr jedoch mit dem Befehl **STEP** andere Werte zuweisen, wie aus den folgenden Beispielen hervorgeht.

a)
```
100 FOR gerade = 2 TO 10 STEP 2
110 PRINT ! gerade !
120 END FOR gerade
```

2 4 6 8 10 wird ausgegeben.

b)
```
100 FOR abnehmend = 9 TO 1 STEP -1
110 PRINT ! abnehmend !
120 END FOR abnehmend
```

9 8 7 6 5 4 3 2 1 wird ausgegeben.

Außerdem können Schleifen verschachtelt werden. Möglicherweise sind Sie schon mit verschachtelten **FOR**-Schleifen vertraut. So werden beispielsweise mit dem folgenden Programm vier Reihen mit zehn Kreuzen ausgegeben:

```
100 REMark Kreuze
110 FOR reihe = 1 TO 4
120   PRINT 'Reihe Nummer'! Reihe
130   FOR kreuz = 1 TO 10
140     PRINT ! ''x'' !
150   END FOR kreuz
160   PRINT
170 PRINT \ 'Ende der Reihe Nummer' ! reihe
180 END FOR reihe
```

Ausgabe:

```
Reihe Nummer 1
X X X X X X X X X X
Ende der Reihe Nummer 1
Reihe Nummer 2
X X X X X X X X X X
Ende der Reihe Nummer 2
Reihe Nummer 3
X X X X X X X X X X
Ende der Reihe Nummer 3
Reihe Nummer 4
X X X X X X X X X X
Ende der Reihe Nummer 4
```

Der große Vorteil von SuperBASIC besteht darin, daß es über Strukturen für alle Zwecke und nicht nur für **FOR**-Schleifen verfügt, und daß alle ineinander verschachtelt werden und somit den Erfordernissen einer Aufgabe angepaßt werden können.

Wir können eine **REPeat**-Schleife in eine **FOR**-Schleife setzen. Mit dem folgenden Programm werden die Augenzahlen von zwei Würfeln in jeder Reihe erzeugt, bis eine 7 geworfen wird.

```
100 REMark Würfel Reihen
110 FOR reihe = 1 TO 4
120   PRINT 'Reihe Nummer' ! reihe
130   REPeat werfen
140     LET augenzahl1 = RND(1 TO 6)
150     LET augenzahl2 = RND(1 TO 6)
160     LET ergebnis = augenzahl1 + augenzahl2
170     PRINT ! ergebnis !
180     IF ergebnis = 7 THEN EXIT werfen
190   END REPeat werfen
200   PRINT\ 'Ende der Reihe Nummer ! reihe
210 END FOR reihe
```

Muster-Ausgabe:

```
Reihe Nummer 1
8 11 6 3 7
Ende der Reihe Nummer 1
Reihe Nummer 2
4 6 2 9 4 5 12 7
Ende der Reihe Nummer 2
Reihe Nummer 3
Ende der Reihe Nummer 3
Reihe Nummer 4
6 2 4 9 9 7
Ende der Reihe Nummer 4
```

Die dritte Schleifenfunktion bei SuperBASIC ermöglicht eine größere Flexibilität bei dem Wertebereich in einer **FOR**-Schleife. Mit dem folgenden Programm wird dies dargelegt, indem alle teilbaren Zahlen von 1 bis 20 ausgedruckt werden. (Eine teilbare Zahl kann auch durch andere Zahlen als 1 und sich selbst dividiert werden.)

```
100 REMark Teilbare zahlen
110 FOR zahl = 4,6,8 TO 10,12,14 TO 16,18,20
120   PRINT ! zahl !
130 END FOR zahl
```

Die Schleifen werden in einem der folgenden Abschnitte noch näher erläutert. Mit den oben beschriebenen Funktionen können jedoch die meisten gebräuchlichen Situationen gemeistert werden.

ENTSCHEIDUNGEN

Sie werden die einfache Entscheidung:

```
IF augenzahl = 6 THEN EXIT werfen
```

bemekrt haben. Sie ist bei den meisten BASIC-Versionen verfügbar. SuperBASIC bietet jedoch Erweiterungen zu dieser Struktur, sowie eine vollständig neue Struktur für die Handhabung von Situationen mit mehr als zwei alternativen Möglichkeiten.

Möglicherweise mögen Sie die folgenden langen Formen von **IF ... THEN**. Sie erläutern sich selbst.

a)
```
100 REMark Lange Version IF...END IF
110 LET sonnig = RND(0 TO 1)
120 IF sonnig THEN
130   PRINT ''Setzen Sie die Sonnenbrille auf''
140   PRINT ''Machen Sie einen Spaziergang''
150 END IF
```

b)
```
100 REMark Lange Version IF...ELSE...END IF
110 LET sonnig = RND(0 TO 1)
120 IF sonnig THEN
130   PRINT 'Setzen Sie sich eine Sonnenbrille auf'
140   PRINT 'Machen Sie einen Spaziergang'
150 ELSE
160   PRINT 'Ziehen Sie sich einen Mantel an'
170   PRINT 'Gehen Sie ins Kino'
180 END IF
```

THEN kann in der langen Form entfallen und kann in der kurzen Form durch einen Doppelpunkt ersetzt werden. Die langen Entscheidungs-Strukturen können ähnlich wie Schleifen, verschachtelt werden. In sie können auch andere Strukturen gesetzt werden. Steht eine einzige Variable an der Stelle, an der Sie eine Bedingung erwarten, so wird der Wert Null als falsch und andere Werte als wahr betrachtet.

UNTERPROGRAMME UND PROZEDUREN

Die meisten BASIC-Versionen verfügen über eine **GOSUB**-Anweisung. Mit ihr kann das Programm zu einem Unterprogramm (Subroutine) verzweigen und anschließend zu der Stelle zurückkehren, von der aus das Unterprogramm aufgerufen wurde. Die **GOSUB**-Anweisung ist unter verschiedenen Gesichtspunkten nicht zufriedenstellend. SuperBASIC bietet benannte Prozeduren mit einigen sehr nützlichen Funktionen.

Betrachten Sie die folgenden Programme, die beide ein grünes 'Quadrat' mit einer Seitenlänge von 50 Pixel-Einheiten bei Position 200, 100 auf einem roten Hintergrund erzeugen.

(a) Benutzung von **GOSUB**:

```
100 LET farbe = 4 : hintergrund = 2
110 LET waag = 20
120 LET senk= 100
130 LET seite = 50
140 GO SUB 170
150 PRINT 'ENDE'
160 STOP
170 REMark Unterprogramm um Quadrat zu zeichnen
180 PAPER hintergrund : CLS
190 BLOCK seite, seite, waag, senk, farbe
200 RETurn
```

(b) Benutzung einer Prozedur mit Parametern:

```
100 quadrat 4, 50, 20, 100, 2
110 PRINT 'ENDE'
120 DEFine PROCedure quadrat(farbe,seite,waag,senk,
             hintergrund)
130   PAPER hintergrund : CLS
140   BLOCK seite, seite, waag, senk, farbe
150 END DEFine
```

Bei dem ersten Programm werden die Werte von *farbe, waagerecht, senkrecht, Seite* mit **LET**-Anweisungen festgelegt, bevor die **GOSUB**-Anweisung zu den Zeilen 180 und 190 verzweigt. Danach wird das Programm durch die **RETURN**-Anweisung nach der GOSUB-Anweisung fortgesetzt.

In dem zweiten Programm werden die Werte in der ersten Zeile als Parameter in dem Prozeduraufruf, *Quadrat*, angegeben, mit dem die Ausführung der Prozedur veranlaßt wird. Gleichzeitig werden die benötigten Werte geliefert.

In ihrer einfachsten Form hat eine Prozedur keine Parameter. Schon in dieser Form besitzt die Prozedur Vorteile gegenüber **GOSUB**, da sie einwandfrei benannt und vom übrigen Programm in einer selbständigen Einheit getrennt ist.

Die Leistungsfähigkeit der Prozeduren wird deutlicher, je größer die Programme werden. Je umfangreicher ein Programm wird, desto deutlicher wird der Gewinn an Übersichtlichkeit für den Programmierer.

Beispiele

Die folgenden Beispiele benutzen das Vokabular und die Syntax von SuperBASIC, die in diesem und anderen Abschnitten schon besprochen wurden. Sie bilden eine Grundlage für den zweiten Teil dieses Handbuchs.

Die Buchstaben eines Palindroms (eines Satzes, der, vorwärts wie rückwärts gelesen einen Sinn ergibt) werden als einzelne Elemente in **DATA**-Anweisungen angegeben. Das letzte Element ist ein Sternchen. Dabei gehen Sie davon aus, daß Sie die Anzahl von Buchstaben in dem Palindrom nicht kennen. Nun werden die Buchstaben mit **READ** in eine Tabelle eingelesen und rückwärts ausgedruckt. Einige Palindrome wie 'EIN NEGER MIT GAZELLE ZAGT IM REGEN NIE' können nur benutzt werden, wenn Leerzeichen und Satzzeichen ignoriert werden. Das hier benutzte Beispiel funktioniert einwandfrei.

```
100 REMark Palindrome
110 DIM text$(30)
120 LET text$ = FILL$ (' ',30)
130 LET zahl = 30
140 REPeat holt_buchstaben
150   READ zeichen$
160   IF zeichen$ = '*' THEN EXIT holt_buchstaben
170   LET zahl = zahl-1
180 LET text$(zahl) = zeichen$
190 END REPeat holt_buchstaben
200 PRINT text$
210 DATA 'R','E','L','I','E','F','P','F','E','I',
         'L' ,'E' ,'R','*'
```

Bei dem folgenden Programm können Zahlen in dem Bereich von 1 bis 3999 eingegeben werden. Sie werden in die entsprechenden römischen Zahlen umgewandelt. Damit wird nicht die eleganteste Form erzeugt. Statt IIII wird IV ausgegeben.

```
100 REMark Römische Zahlen
110 INPUT zahl
120 RESTORE 210
130 FOR einlesen = 1 TO 7
140 READ buchstabe$, wert
150   REPeat ausgabe
160     IF zahl < wert : EXIT ausgabe
170     PRINT buchstabe$;
180     LET zahl = zahl - wert
190   END REPeat ausgabe
200 END FOR einlesen
210 DATA 'M',1000,'D',500,'C',100,'L',50,
      'X',10,'V',5,'I',1
```

Arbeiten Sie die letzten Beispiele bitte sorgfältig durch. Führen Sie 'Trockenläufe' aus, bis Sie sicher sind, daß Sie die Beispiele auch wirklich verstanden haben.

SCHLUSS-BEMERKUNG

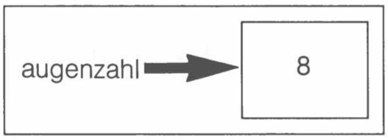
Mit SuperBASIC stehen Ihnen alle Möglichkeiten des strukturierten Programmierens offen. Sie können also einzelne Teilaufgaben getrennt behandeln und sie dann nahtlos zu einem Programm zusammenfügen.

SuperBASIC stellt Ihnen viele Hilfsmittel zur Verfügung, mit denen Sie sich die Arbeit vereinfachen können. Alle Einzelheiten sind in den Abschnitten *Befehle* und *Begriffe* erläutert. Für einen ersten Zugang ist es aber sicher leichter, wenn Sie sich erst einmal einen groben Überblick verschaffen, indem Sie die folgenden Kapitel durcharbeiten. Wenn Sie dabei an weiteren Einzelheiten interessiert sind, dann können Sie diese bei den *Befehlen* oder *Begriffen* nachschlagen. Teile der folgenden Kapitel, die Ihnen im Augenblick weniger wichtig erscheinen, können Sie beim ersten Durcharbeiten getrost überspringen.

KAPITEL 9
DATENTYPEN VARIABLEN UND NAMEN

Sie werden festgestellt haben, daß ein Programm (eine Folge von Anweisungen) normalerweise Daten zur Bearbeitung benötigt (Eingabe) und bestimmte Ergebnisse liefert (Ausgabe). Außerdem wissen Sie, daß es interne Möglichkeiten zur Speicherung dieser Daten gibt. Sie können sich vereinfacht vorstellen, daß die Daten in Ablagefächern gespeichert werden, denen Sie sinnvolle Namen zuteilen. Muß beispielsweise ein Zahl gespeichert werden, die die Augenzahl bei einem simulierten Würfelspiel darstellt, so definieren Sie ein Ablagefach namens Augenzahl, das beispielsweise eine Zahl wie 8 enthalten kann.

Intern sind die Ablagefächer numeriert. Das System unterhält ein Verzeichnis, in dem die Namen mit bestimmten numerierten Ablagefächern verknüpft sind. So weist beispielsweise der Name, *augenzahl*, (mit Hilfe des internen Verzeichnisses) auf das ihm zugeordnete Ablagefach.



Die ganze Anordnung wird als **Variable** bezeichnet.

Hier sehen Sie nun das Wort *augenzahl*. Dieses Wort, *augenzahl*, wird als Name bezeichnet. Ihn sehen wir und er kennzeichnet das benötigte Konzept, in diesem Fall das Ergebnis, 8, aus dem Werfen zweier Würfel. Beim Programmieren brauchen wir nur an den Namen zu denken und an die Bedeutung, die er für das Programm hat. Wenn das Programm ausgeführt wird entspricht ihm, ab der ersten Zuweisung, immer ein bestimmter Wert. Dieser Wert kann sich, während das Programm läuft, natürlich häufig verändern.

Es gibt drei einfache Datentypen: Gleitkomma-, Ganzzahlige- und Stringdatentypen. Der Datentyp logisch ist kein eigenständiger Typ. Jeder numerische Typ kann auch als logischer Typ interpretiert werden. Als logischer Typ hat er den Wert *falsch* wenn der numerische Wert Null ist. Ansonsten hat er den Wert *wahr.* Wir sprechen von Datentypen und nicht von Variablentypen, da die Daten alleine als Wert einer Variablen auftreten können, z.B. 3.4 oder 'blauer Hut'.

NAMEN UND VARIABLEN

1. Ein SuperBASIC-Name muß mit einem Buchstaben beginnen und besteht aus einer Folge von:
   - Groß- oder Kleinbuchstaben
   - Unterstreichungszeichen.
2. Ein Name kann eine Länge von maximal 255 Zeichen aufweisen. Praktisch kann er also beliebig lang sein.
3. Ein SuperBASIC-Befehl darf nicht als Name benutzt werden.
4. Der Name einer ganzzahligen Variablen hat am Ende ein **%**-Zeichen.
5. Der Name einer String-Variablen hat am Ende ein **$**-Zeichen
6. Die Symbole **%** und **$** dürfen in anderen Namen nicht benutzt werden.
7. Wählen Sie Namen so aus, daß aus ihnen ihre Bedeutung im Programm ersichtlich ist. Für SuperBASIC haben Namen allerdings keine weitere Bedeutung, als bestimmte Dinge in einem Programm zu benennen.

GLEITKOMMA-VARIABLEN

Nachfolgend einige Beispiele für die Benutzung von Gleitkommavariablen:

```
100 LET tage = 24
110 LET umsatz = 3649.84
120 LET tagesumsatz = umsatz/tage
130 PRINT tagesumsatz
```

Der Wert einer Gleitkommavariablen kann in folgendem Bereich liegen:

$\pm 10^{-615}$ bis $\pm 10^{615}$ mit 8 signifikanten Ziffern.

Angenommen, in dem obigen Programm beläuft sich der Umsatz ausnahmsweise nur auf 3 Pfennig. Hierzu wird Zeile 110 folgendermaßen geändert:

```
110 LET umsatz = 0.03
```

Das System ändert dies in:

```
110 LET umsatz = 3E-2
```

Um dies zu interpretieren, beginnen Sie mit 3 oder 3.0 und verschieben Sie den Dezimalpunkt um zwei Stellen nach links. Daraus ergibt sich, daß:

3E–2 = 0.03

ist.

Nach Ausführung des Programms beläuft sich der durchschnittliche Tagesumsatz auf:

1.25E–3, was gleichbedeutend mit 0.00125 ist.

Zahlen mit einem E sind in der *Exponentialform* angegeben:

(Mantisse) E (Exponent) = (Mantisse) x 10 hoch (Exponent)

GANZZAHLIGE VARIABLEN

Ganzzahlige Variablen können Werte im Bereich von **–32678** bis **32768** aufweisen. Nachfolgend einige Beispiele für gültige Namen von ganzzahligen Variablen. Sie müssen mit % enden.

```
LET zahl% = 10
LET zwei_max% = RND(10)
LET zahl_3% = 3
```

Der einzige Nachteil der ganzzahligen Variablen liegt in dem etwas irreführenden **%**-Symbol am Ende des Namens. Dieses Symbol hat nichts mit dem Prozentsatz zu tun. Hier handelt es sich einfach um ein bequemes Symbol, das angehängt wird, um anzugeben, daß es sich bei den Variablen um eine ganze Zahl handelt.

NUMERISCHE FUNKTIONEN

Die Benutzung einer Funktion kann in etwa mit dem Zubereiten eines Omelettes verglichen werden. Sie schlagen ein Ei in die Pfanne, das dann nach bestimmten Regeln (dem Rezept) zubereitet wird und schließlich ein Omelette ergibt. So nimmt beispielsweise die Funktion **INT** einen beliebigen Ausdruck als Eingabe und gibt den ganzzahligen Teil zurück. Die Angaben, die in eine Funktion eingegeben werden, werden als Parameter oder Argument bezeichnet. Sie können:

```
PRINT INT(5.6)
```

schreiben und 5 würde ausgegeben. Wir sagen, daß 5.6 der Parameter ist und die Funktion den Wert 5 zurückgibt. Eine Funktion kann mehr als einen Parameter benötigen. Die Funktion:

```
RND(1 TO 6)
```

kennen Sie schon. Hier handelt es sich um eine Funktion mit zwei Parametern. Funktionen geben jedoch genau einen Wert zurück. Dies muß so sein, da Sie Funktionen in Ausdrücke setzen können. Zum Beispiel erzeugt:

```
PRINT 2 * INT(5.6)
```

die Ausgabe 10. Eine wichtige Eigenschaft der Funktionen ist, daß sie in Ausdrücken benutzt werden können. Daraus ergibt sich, daß sie einen einzelnen Wert zurückgeben müssen, der dann in dem Ausdruck benutzt wird. **INT** und **RND** sind Systemfunktionen; sie werden mit dem System geliefert. Später wird jedoch beschrieben, wie Sie eigene Funktionen selbst schreiben können.

Die folgenden Beispiele zeigen gebräuchliche Anwendungen der **INT**-Funktion.

```
100 REMark aufrunden
110 INPUT Dezimalzahl
120 PRINT INT(Dezimalzahl + 0.5)
```

In diesem Beispiel geben Sie einen Dezimalbruch ein und die Ausgabe wird gerundet. So wird 4.7 zu 5, während 4.3 zu 4 wird.

Sie können dasselbe Ergebnis mit einer ganzzahligen Variablen und der Datentypumwandlung erzielen.

Trigonometrische Funktionen werden später in diesem Abschnitt behandelt. Einige andere gebräuchliche numerische Funktionen werden in der folgenden Liste angeführt.

Funktion	Auswirkung	Beispiele	Zurückgegebene Werte
ABS	Absoluter Wert oder	ABS(7)	7
	Wert ohne Vorzeichen	ABS(–4.3)	4.3
INT	Ganzzahliger Teil einer	INT(2.4)	2
	Gleitkommazahl	INT(0.4)	0
		INT(–2.7)	–3
SQRT	Quadratwurzel	SQRT(2)	1.414214
		SQRT(16)	4
		SQRT(2.6)	1.612452

Es gibt eine einfach verständliche Art zur Berechnung von Quadratwurzeln. Um die Quadtratwurzel von 8 zu zu berechnen, stellen Sie zuerst eine Schätzung an. Es spielt keine Rolle, wie schlecht diese Schätzung ist. Angenommen, Sie nehmen als erste Schätzung einfach die Hälfte von 8, d.h. 4.

Da 4 größer ist als die Quadratwurzel von 8, muß 8/4 kleiner sein. Das umgekehrte ist ebenfalls richtig. Hätten Sie 2 geschätzt, was kleiner ist als die Quadratwurzel, so muß 8/2 größer sein.

Nehmen wir also eine beliebige Schätzung und berechnen wir Zahl/Schätzung, so erhalten wir zwei Zahlen, eine zu kleine und eine zu große. Nun nehmen wir den Mittelwert dieser beiden Zahlen als nächste Näherung und kommen so der Lösung näher.

Dieses Verfahren wird wiederholt, bis aufeinanderfolgende Näherungen so nahe beieinander sind, daß es praktisch keinen Unterschied mehr gibt.

```
100 REMark Quadratwurzeln
110 LET zahl = 8
120 LET ungef = zahl/2
130 REPeat wurzel
140   LET neuwert = (ungef + zahl/ungef) /2
150   IF neuwert == ungef THEN EXIT wurzel
160   LET ungef = neuwert
170 END REPeat wurzel
180 PRINT 'Die Quadratwurzel von' !zahl ! 'ist' ! neuwert
```

Ausgabe:

```
Die Quadratwurzel von 8 ist 2.828427
```

Hier wird darauf hingewiesen, daß die bedingte **EXIT**-Anweisung aus der Schleife in der Mitte stehen muß. Die üblichen Strukturen werden mit dieser Situation nicht so gut fertig, wie SuperBASIC.

Das == Zeichen in Zeile 150 bedeutet "annähernd gleich", d.h. innerhalb einer Toleranz von .0000001 relativ zu den verglichenen Werten.

NUMERISCHE OPERATIONEN

SuperBASIC läßt die üblichen mathematischen Operationen zu. Sie werden feststellen, daß sie Funktionen mit jeweils genau zwei Operanden entsprechen. In diesen Fällen wird üblicherweise jeweils ein Operand rechts und links neben das Symbol gestellt. Gelegentlich wird die Operation durch ein bekanntes Symbol wie beispielsweise + oder * gekennzeichnet. Gelegentlich wird die Operation auch durch ein Schlüsselwort wie **DIV** oder **MOD** gekennzeichnet, dies macht jedoch keinen großen Unterschied. Numerische Operationen haben eine Prioritätsfolge. So ist beispielsweise das Ergebnis von:

```
PRINT 7 + 3*2
```

13, da die Multiplikation eine höhere Priorität hat. Bei:

```
PRINT (7 + 3)*2
```

wird jedoch 20 ausgegeben, da die Klammern die übliche Priorität außer Kraft setzen. Nähere Ausführungen über die Priorität können Sie in dem Abschnitt *Begriffe* nachschlagen. Die Operationen, mit denen wir hier arbeiten, haben folgende Prioritäten:

Höchste Priorität	–	Potenzieren
		Multiplikation und Division (einschließlich DIV, MOD)
Niedrigste Priorität	–	Addition und Subtraktion.

Die Symbole + und – können auch mit nur einem Operanden benutzt werden, der einfach positiv oder negativ angibt. Auf diese Art und Weise benutzte Symbole haben die höchste Priorität von allen und können nur durch Benutzung von Klammern außer Kraft gesetzt werden.

Haben schließlich zwei Symbole dieselbe Priorität, so wird die am weitesten links stehende Operation als erste ausgeführt. Bei:

```
PRINT 7-2 + 5
```

wird also die Subtraktion vor der Addition ausgeführt. Dies kann von Bedeutung sein, wenn Sie später einmal mit sehr großen oder sehr kleinen Zahlen arbeiten.

Operation	Symbol	Beispiele	Ergebnisse	Hinweise
Addition	+	7+6.6	13.6	
Subtraktion	–	7–6.6	0.4	
Multiplikation	*	3*2.1	6.3	
		2.1*(–3)	–6.3	
Division	/	7/2 –17/5	–3.4	Keine Division durch Null
Potenzierung	^	4^1.5	8	
Ganzzahlige Division	DIV	–8 DIV 2 7 DIV 2	–4 3	Nur ganze Zahlen Keine Division durch Null
Modul	MOD	13 MOD 5 21 MOD 7 –17 MOD 8	3 0 7	

Modul gibt den Rest einer ganzzahligen Division zurück. Jeder Versuch einer Division durch Null führt zu einem Fehler und beendet die Programmausführung.

NUMERISCHE AUSDRÜCKE

Genau genommen ist ein numerischer Ausdruck ein Ausdruck, bei dessen Auswertung sich eine Zahl ergibt. Nun gibt es verschiedene Möglichkeiten, die hier besprochen werden sollen. In SuperBASIC können Sie, falls gewünscht, kompliziert zusammengesetzte Operationen ausführen. Sie können jedoch auch einfache Dinge auf einfache Art und Weise ausführen. In diesem Abschnitt konzentrieren wir uns auf die übliche direkte Benutzung der mathematischen Funktionen.

Im wesentlichen sind numerische Ausdrücke in SuperBASIC gleichbedeutend mit numerischen Ausdrücken in der Mathematik. Der ganze Ausdruck muß jedoch in eine Folge von Zeichen gebracht werden.

$$\frac{5 + 3}{6 - 4}$$

wird in SuperBASIC (oder anderen BASIC-Versionen) zu:

(5 + 3)/(6 – 4)

Aus der Algebra kennen wir einen Ausdruck für eine Lösung einer quadratischen Gleichung:

$$ax^2 + bx + c = 0$$

Eine Lösung in mathematischer Schreibweise sieht folgendermaßen aus:

$$x = \frac{-b + \sqrt{b^2-4ac}}{2a}$$

Beginnen wir mit der Gleichung:

$$2x^2 - 3x + 1 = 0$$

so wird mit folgendem Programm eine Lösung gefunden.

Beispiel 1

```
100 READ a,b,c
110 PRINT 'Die Wurzel ist' ! (-b + SQRT(b^2 - 4*a*c))/(2*a)
120 DATA 2,-3,1
```

Beispiel 2 Bei Aufgaben, in denen ein Kartenspiel simuliert werden soll, können den Karten wie folgt die Zahlen 1 bis 52 zugewiesen werden:

1 bis 13	As, Zwei .. Herz-König
14 bis 26	As, Zwei .. Kreuz-König
27 bis 39	As, Zwei .. Karo-König
40 bis 52	As, Zwei .. Pik-König

Eine bestimmte Karte kann wie folgt gekennzeichnet werden:

```
100 REM Spielkarten_Erkennung
110 LET karte = 23
120 LET farbe = (karte-1) DIV 13
130 LET wert = karte MOD 13
140 IF wert = 0 THEN LET wert = 13
150 IF farbe = 0 THEN PRINT ''Herz'';
160 IF farbe = 1 THEN PRINT ''Kreuz'';
170 IF farbe = 2 THEN PRINT ''Karo'';
180 IF farbe = 3 THEN PRINT ''Pik'';
190 IF wert = 1 THEN PRINT ''Ass'';
200 IF wert >= 2 AND wert >= 10 THEN PRINT wert;
210 IF wert = 11 THEN PRINT ''Bube''
220 IF wert = 12 THEN PRINT ''Dame''
230 IF wert = 13 THEN PRINT ''König''
```

In diesem Programm gibt es einige neue Ideen. Sie stehen in Zeile 200. Diese Zeile bedeutet eindeutig, daß die Zahl nur ausgedruckt wird, wenn zwei logische Anweisungen wahr sind, d.h.:

Wert ist größer oder gleich 2 **UND** Wert ist kleiner oder gleich 10.

Bei Karten außerhalb dieses Bereichs handelt es sich entweder um Asse oder um "Bilder", die anders behandelt werden müssen.

Beachten Sie die Benutzung des ! in der PRINT-Anweisung für ein Leerzeichen. Mit dem ; wird gewährleistet, daß die Ausgabe auf derselben Zeile fortgesetzt wird.

Es gibt zwei Gruppen von mathematischen Funktionen, mit denen wir uns hier noch nicht befaßt haben, und zwar die trigonometrischen und logarithmischen Funktionen. Die ersteren werden unter Umständen bei der Organisation der Bildschirmanzeigen benötigt. Diese Funktionstypen werden ebenfalls voll in dem Kapitel *Begriffe* erläutert.

LOGISCHE VARIABLEN

Genau genommen läßt SuperBASIC keine logischen Variablen zu, Sie können jedoch andere Variablen als logische Variablen benutzen. So können Sie beispielsweise das folgende Programm ausführen:

```
100 REMark Logische Variable
110 LET hunger = 1
120 IF hunger THEN PRINT ''Essen Sie ein Brötchen''
```

In Zeile 120 erwarten Sie einen logischen Ausdruck, die numerische Variable, hunger, steht jedoch alleine. Das System interpretiert den Wert von Hunger als wahr und nimmt folgende Ausgabe vor:

```
Essen Sie ein Brötchen
```

Steht in Zeile 110:

```
LET hunger = 0
```

so würde nichts ausgegeben. Das System interpretiert Null als falsch und alle anderen Werte als wahr. Dies ist nützlich, Sie können jedoch die numerische Eigenschaft von Hunger verdecken, indem Sie folgendes Programm schreiben:

```
100 REMark Logische Variable
110 LET wahr = 1 : falsch = 0
120 LET hunger = wahr
130 IF hunger THEN PRINT ''Essen Sie ein Brötchen''
```

STRING-VARIABLEN

Über die Verarbeitung von Strings und String-Variablen ist sehr viel zu sagen. Deshalb werden sie in einem separaten Kapitel erläutert.

AUFGABEN ZU KAPITEL 9

1. Ein reicher Ölhändler spielt, indem er eine Münze folgendermaßen wirft. Bei Kopf erhält er 1. Bei Zahl wirft er die Münze erneut, der Einsatz wird jedoch verdoppelt. Dies wird wiederholt, so daß sich folgende Punktzahlen ergeben.

WURF	1	2	3	4	5	6	7
ERGEBNIS	1	2	4	8	16	32	64

   Simulieren Sie das Spiel und entscheiden Sie, welches ein fairer Einsatz für ein solches Spiel ist:

   (a) wenn der Spieler auf maximal sieben Würfe pro Spiel beschränkt ist,

   (b) wenn es für die Anzahl von Würfen keine Höchstzahl gibt.

2. Karl und Otto einigen sich auf folgendes Spiel. Bei einem bestimmten Signal wird jeder Einsatz in zwei Teile geteilt, wobei die eine Hälfte dem anderen Spieler übergeben wird. Jeder Spieler teilt dann seine neue Gesamtsumme und gibt die Hälfte dem Partner. Legen Sie dar, was im Laufe des Spiels geschieht, wenn Karl mit 16 Pfennig und Otto mit 64 Pfennig beginnt.

3. Was geschieht, wenn das Spiel so geändert wird, daß jeder Spieler dem anderen einen Betrag übergibt, der gleich der Hälfte dessen ist, was der andere Spieler besitzt?

4. Schreiben Sie ein Programm, mit dem zufällige aus drei Buchstaben bestehende Wörter, wobei aus A, B, C, D gewählt wird, erzeugt werden und diese ausdruckt, bis 'BAD' angezeigt wird.

5. Ändern Sie das letzte Programm, so daß es beendet wird, wenn ein richtiges aus drei Buchstaben bestehendes Wort angezeigt wird.

KAPITEL 10
LOGIK

Nachdem Sie die vorhergehenden Abschnitte gelesen haben, werden Sie wahrscheinlich mit uns übereinstimmen, daß Wiederholung, Entscheidungsfällung und Aufteilung von Aufgaben in Unteraufgaben wichtige Konzepte bei der Problemanalyse, dem Programmaufbau und der Codierung von Programmen darstellen. Zwei dieser Konzepte, Schleifen und Entscheidungsfällung, benötigen logische Ausdrücke, wie in den folgenden Programmzeilen:

```
IF ergebnis = 7 THEN EXIT werfen
IF farbe = 3 THEN PRINT ''Pik''
```

Mit der ersten Zeile kann mit **EXIT** aus einer **REPeat**-Schleife gegangen werden. Bei der zweiten Zeile handelt es sich einfach um eine Entscheidung, etwas zu tun oder nicht. Ein mathematischer Ausdruck kann einen von Millionen möglicher numerischer Werte ergeben. Gleichermaßen kann ein String-Ausdruck Millionen möglicher Zeichen-Strings ergeben. Sie werden es seltsam finden, daß logische Ausdrücke, denen große Bedeutung zugewiesen wird, nur zwei verschiedene Ergebnisse haben können: *wahr oder falsch.*

Bei:

```
ergebnis = 7
```

ist dies eindeutig richtig. Entweder ist das Ergebnis gleich 7 oder nicht! Der Ausdruck muß wahr oder falsch sein – vorausgesetzt, er ist nicht bedeutungslos. Möglicherweise kennen Sie den Wert zu einem bestimmten Zeitpunkt nicht, dies ändert sich jedoch im Laufe der Zeit.

Bei Ausdrücken mit den Wörtern **ODER**, **UND**, **NICHT** muß mit etwas Vorsicht vorgegangen werden. Es lohnt sich jedoch durchaus, sich mit diesen Ausdrücken zu befassen – in der Tat sind sie für eine gute Programmierung lebenswichtig. Sie werden sogar mit dem Trend zu anderen Sprachen noch wichtiger, die auf einer genaueren Beschreibung dessen beruhen, was Sie benötigen und nicht was der Computer tun muß.

UND

Das Wort **UND** in SuperBASIC kann mit dem normalen Wort "und" in der deutschen Sprache verglichen werden. Betrachten Sie das folgende Programm.

```
100 REMark AND
110 PRINT ''Geben Sie zwei Werte ein''\''1 für richtig oder
      0 für falsch''
120 INPUT regen, loch_im_dach
130 IF regen AND loch_im_dach THEN PRINT ''Es wird naß im
Haus''
```

Wie im tatsächlichen Leben werden Sie nur naß, wenn es regnet und im Dach ein Loch ist. Sind eine (oder beide) der einfachen logischen Variablen

Regen
Loch_im_Dach

falsch, so ist der zusammengesetzte logische Ausdruck

Regen **UND** *loch_im_Dach*

ebenfalls falsch. Zwei wahre Werte sind erforderlich, damit der ganze Ausdruck wahr wird. Dies ergibt sich aus folgenden Regeln. Nur wenn der zusammengesetzte Ausdruck wahr ist, werden wir naß.

Regen	Loch_im_Dach	Regen UND Loch_im_Dach	Auswirkung
FALSCH	FALSCH	FALSCH	TROCKEN
FALSCH	WAHR	FALSCH	TROCKEN
WAHR	FALSCH	FALSCH	TROCKEN
WAHR	WAHR	WAHR	NASS

Regeln für UND

ODER

Im täglichen Leben wird das Wort 'oder' auf zweierlei Art benutzt. Wir können diese Benutzung von **ODER** darstellen, indem wir uns einen Fußballtrainer vorstellen, der

nach neuen Spielern sucht. Er kann folgende Frage stellen "Können Sie Tore schießen oder Tore halten?". Er könnte damit zufrieden sein, wenn ein Spieler nur eines der beiden Dinge kann, könnte jedoch auch nach einem Spieler suchen, der beide Dinge kann. Ebenso verhält es sich bei der Programmierung: ein zusammengesetzter Ausdruck, in dem **ODER** benutzt wird, ist wahr, wenn eine der beiden der einfachen Bedingungen oder Variablen wahr sind. Geben Sie folgendes Programm ein:

```
100 REMark OR Test
110 PRINT ''Geben Sie zwei Werte ein''\''1 für richtig oder
         0 für falsch''
120 INPUT ''Können Sie Tore schießen ?'', libero
130 INPUT ''Können Sie Tore halten ?'' , torwart
140 IF libero OR torwart THEN PRINT ''Sie sind in der
    Mannschaft''
```

Sie können Die Auswirkungen der verschiedenen Kombinationen anhand der Antworten in der folgenden Tabelle sehen:

Libero	Torwart	Libero ODER Torwart	Auswirkungen
FALSCH	FALSCH	FALSCH	Nicht im Team
FALSCH	WAHR	WAHR	im Team
WAHR	FALSCH	WAHR	im Team
WAHR	WAHR	WAHR	im Team

Regeln für ODER

Wird ein **Inklusives ODER** benutzt, so erzeugt ein wahrer Wert in einer der einfachen Anweisungen einen wahren Wert in dem zusammengesetzten Ausdruck. Würde Franz Beckenbauer die beiden Fragen beantworten müssen, so wären beide einfachen Anweisungen und der zusammengesetzte Ausdruck wahr. Er würde in dem Team spielen.

Wird Null für Falsch und Eins für Wahr geschrieben, so erhalten Sie alle möglichen Kombinationen, wenn Sie mit Binärzahlen zählen:

00
01
10
11

NICHT

Das Wort **NICHT** hat eine offensichtliche Bedeutung.

NICHT *wahr* ist dasselbe wie *falsch.*
NICHT *falsch* ist dasselbe wie *wahr*

Allerdings müssen Sie vorsichtig vorgehen. Angenommen, Sie haben ein rotes Dreieck und definieren es folgendermaßen:

NICHT *rot* UND *Quadrat*

im Sprachgebrauch kann dies zweideutig sein.

Meinen Sie:

(NICHT *rot*) UND *Quadrat*

so ist der Ausdruck für ein rotes Dreieck falsch.

Meinen Sie:

NICHT (*rot* UND *Quadrat*),

so ist der ganze Ausdruck für ein rotes Dreieck wahr. In der Programmierung muß es Regeln geben, die Zweideutigkeiten ausschließen. Diese Regel besteht darin, daß **NICHT** gegenüber **UND** vorrangig ist. So ist die Interpretation:

(NICHT *rot*) UND *Quadrat*

richtig. Sie ist gleichbedeutend mit:

NICHT *rot* UND *Quadrat*

Für die andere Interpretation müssen Klammern benutzt werden. Wird ein komplexer logischer Ausdruck benötigt, so werden am besten Klammern und **NICHT** benutzt, wenn dadurch auf natürliche Weise angegeben wird, was Sie wünschen. Gegebenenfalls können Sie jdoch die Klammern stets weglassen, indem Sie folgende Regeln beachten (Sie werden Augustus De Morgan zugeschrieben).

> **NICHT** (a **UND** b)
> ist gleichbedeutend mit **NICHT** a **ODER NICHT** b
>
> **NICHT** (a **ODER** b)
> ist gleichbedeutend mit **NICHT** a **UND NICHT** b

Zum Beispiel:

> **NICHT** (*groß* **UND** *schlank*)
> ist gleichbedeutend mit
> **NICHT** *groß* **ODER NICHT** *schlank*
>
> **NICHT** (*Hunger* **ODER** *Durst*)
> ist gleichbedeutend mit
> **NICHT** *Hunger* **UND NICHT** *Durst*

Testen Sie dies, indem Sie folgendes Programm eingeben:

```
100 REMark NOT und Klammern
110 PRINT ''Geben Sie zwei Werte ein''\''1 für richtig oder
        0 für falsch''
120 INPUT ''Groß'';groß
130 INPUT ''Blond''; blond
140 IF NOT (groß AND blond) THEN PRINT ''Erstes''
150 IF NOT groß OR NOT blond THEN PRINT ''Zweites''
```

Gleichgültig, welche Zahlenkombination Sie auch eingeben, die Ausgabe besteht stets aus zwei Wörtern oder keinem Wort aber niemals aus einem Wort. Daraus ergibt sich, daß die beiden zusammengesetzten logischen Ausdrücke gleichbedeutend sind.

XOR-EXKLUSIVES ODER

Angenommen, ein Golf Profi sucht nach einem Assistenten, der entweder das Geschäft führen oder Golfunterricht geben kann. Meldet sich ein Bewerber, der beides kann, so bekommt er den Job vielleicht nicht, weil der Profi Angst hat, daß ihn ein solch fähiger Assistent beiseite drängt. Er würde einen guten Golfer akzeptieren, der das Geschäft nicht führen kann oder würde einen schlechten Golfer akzeptieren, der das Geschäft führen kann. Hier handelt es sich um eine Exklusive ODER Situation: eine der Fähigkeiten wird akzeptiert, jedoch nicht beide. Mit dem folgenden Programm würden Bewerber getestet:

```
100 REMark XOR Test
110 PRINT ''Geben Sie 1 für richtig oder 0 für falsch ein''
120 INPUT ''Können Sie ein Geschäft führen ?'' , verkaufen
130 INPUT ''Können Sie Golf lehren ?'' , lehren
140 IF verkaufen XOR lehren THEN PRINT ''Eingestellt''
```

Die einzigen Antwortkombinationen, die zu der Ausgabe "Eingestellt" führen, sind (0 und 1) oder (1 und 0). Die Regeln für **XOR** werden nachfolgend angegeben.

Kann Geschäft führen	Kann lehren	Geschäft XOR lehren	Auswirkung
FALSCH	FALSCH	FALSCH	Kein Job
FALSCH	WAHR	WAHR	Erhält den Job
WAHR	FALSCH	WAHR	Erhält den Job
WAHR	WAHR	FALSCH	Kein Job

Regeln für XOR

PRIORITÄTEN

Die Prioritätsfolge für die logischen Operatoren lautet (höchste Priorität zuerst):

NICHT
UND
ODER, XOR

So ist beispielsweise der Ausdruck:

reich ODER *groß* UND *schlank*

gleichbedeutend mit:

reich ODER (*groß* UND *schlank*)

Die UND-Operation wird als erstes ausgeführt. Um zu beweisen, daß die beiden logischen Ausdrücke dieselbe Auswirkung haben, führen Sie folgendes Programm aus:

```
100 REMark Vorrangstufen
110 PRINT ''Geben Sie 3 Werte ein''\''1 für richtig oder
        0 für falsch''
120 INPUT reich , groß , blond
130 IF reich OR groß AND blond THEN PRINT ''Ja''
140 IF reich OR (groß AND blond) THEN PRINT ''OK''
```

Gleichgültig, welche Kombination von drei Nullen oder Einsen in Zeile 120 eingegeben werden, als Ausgabe wird stets entweder nichts oder:

```
JA
OK
```

ausgegeben. Sie können sich vergewissern, daß Sie sämtliche Möglichkeiten getestet haben, indem Sie Daten eingeben, die acht dreistellige Binärzahlen von 000 bis 111 darstellen:

000 001 010 011 100 101 110 111

AUFGABEN ZU KAPITEL 10

1. Setzen Sie zehn Zahlen in eine DATA-Anweisung. Lesen Sie jede Zahl mit READ. Ist sie größer als 20, so drucken Sie sie aus.
2. Testen Sie sämtliche Zahlen von 1 bis 100 und drucken Sie nur die Zahlen aus, die durch 7 teilbar oder Quadratzahlen sind.
3. Spielzeug wird als Sicher (S), Unsicher (U), Teuer (T), Billig (B) für Mädchen (M), Jungen (J) oder Alle (A) eingestuft. Mit drei Buchstaben werden die Eigenschaften jedes Spielzeugs codiert. Setzen Sie fünf derartige Trios in eine DATA-Anweisung. Danach durchsuchen Sie die Spielzeuge, wobei nur die ausgedruckt werden, die sicher und für Mädchen geeignet sind.
4. Ändern Sie Programm 3 so, daß nur die Spielzeuge ausgedruckt werden, die teuer und unsicher sind.
5. Ändern Sie Programm 3 so, daß die Spielzeuge ausgedruckt werden, die sicher, nicht teuer und für alle geeignet sind.

KAPITEL 11

VERARBEITUNG VON TEXT - STRINGS

Sie haben schon String-Variablen zur Speicherung von Zeichen-Strings benutzt und wissen, daß die Regeln für die Verarbeitung von String-Variablen oder String-Konstanten nicht dieselben sind, wie für die Verarbeitung von numerischen Variablen oder numerischen Konstanten. SuperBASIC bietet ein breites Spektrum an Funktionen zur wirksamen Verarbeitung von Zeichen-Strings. Insbesondere das Konzept der String-Aufteilung erweitert und vereinfacht die Verarbeitung von Teilstrings oder Teilen eines String.

ZUWEISUNG VON STRINGS

Der Speicherplatz für String-Variablen wird nach Bedarf von einem Programm zugewiesen. So führen beispielsweise die Zeilen:

```
100 LET wort$ = ''LANG''
110 LET wort$ = ''LÄNGER''
120 PRINT wort$
```

zum Ausdrucken des aus sechs Buchstaben bestehenden Wortes, LÄNGER. Mit der ersten Zeile wird Speicherplatz für vier Buchstaben zugewiesen. Diese Zuweisung wird jedoch von der zweiten Zeile außer Kraft gesetzt, mit der Speicherplatz für sechs Zeichen angefordert wird.

Nun kann jedoch eine String-Variable dimensioniert werden (d. h. es kann Platz für sie reserviert werden). In diesem Fall wird die Maximallänge definiert und die Variable verhält sich wie eine Tabelle.

VERKETTEN VON STRINGS

Bei der Datenverarbeitung wird man Datensätze aus einer Reihe von Quellen erstellen wollen. Angenommen, Sie sind Lehrer und möchten Noten in Mathematik, Geschichte und Englisch für jeden Schüler festlegen. Die Noten werden wie dargestellt in Variablen gespeichert:

mathe$	62	gesch$	56	engl$	71

Als Teil der Aufzeichnung für jeden Schüler möchten Sie vielleicht drei String-Werte in einem sechs Zeichen umfassenden String namens *note$* kombinieren. Hierzu schreiben Sie einfach:

```
LET note$ = mathe$ & gesch$ & engl$
```

Damit haben Sie eine weitere Variable erstellt, wie nachfolgend angegeben:

note$	625671

Denken Sie jedoch daran, daß Sie es hier mit einem Zeichen-String zu tun haben, der zufällig Zahlenzeichen enthält und nicht mit einer numerischen Variablen. Bei SuperBASIC wird das & Symbol dazu benutzt, Strings miteinander zu verketten, während bei einigen BASIC-Versionen zu diesem Zweck das + benutzt wird.

KOPIEREN EINES TEILSTRING

Ein Teilstring ist Bestandteil eines String. Er kann ein einzelnes Zeichen bis zum ganzen String umfassen. Zur Definition des Teilstring müssen Sie die Positionen der erforderlichen Zeichen kennen.

Angenommen, Sie erstellen ein Kinderspiel, bei dem ein Wort aus einer Folge von zufälligen Buchstaben erkannt werden soll. Jeder Buchstabe verfügt über eine interne Nummer – einen Index –, die seiner Position in dem String entspricht. Angenommen, der ganze String ist in der Variablen *wirrwarr* gespeichert und der Schlüssel ist Tier.

Teilstring (6–9)

wirrwarr$	A	Q	W	E	R	E	N	T	E	L	O	P	U
	1	2	3	4	5	6	7	8	9	10	11	12	13

Wie Sie sehen, wird die Antwort mit den Nummern 6 bis 9 definiert, mit denen angegeben wird, an welcher Stelle die Antwort steht. Sie können die Antwort wie nachfolgend dargestellt abstrahieren:

```
100 LET wirrwarr$ = ''aqwerentelopu''
110 LET tier$ = wirrwarr$(6 TO 9)
120 PRINT tier$
```

ERSETZEN EINES TEILSTRING

Angenommen, Sie möchten das Tier nun in eine Gans ändern. Hierzu schreiben Sie zwei zusätzliche Zeilen:

```
130 LET wirrwarr$(6 TO 9) = ''gans''
140 PRINT wirrwarr$
```

Die Ausgabe aus dem ganzen fünfzeiligen Programm lautet:

```
ente
aqwerganslopu
```

Sämtliche String-Variablen sind ursprünglich leer. Sie weisen eine Länge von Null auf. Versuchen Sie, einen String in einen Teilstring zu kopieren, der nicht über genügend Platz verfügt, so kann die Zuweisung unter Umständen nicht von SuperBASIC erkannt werden.

Möchten Sie einen String in einen Teilstring kopieren, so vergewissern Sie sich am besten zuerst, ob der Bestimmungs-String lang genug ist, indem Sie ihn erst einmal mit Leerzeichen auffüllen.

```
100 LET fach$ = ''DEUTSCHES MATHE COMPUTERN''
110 LET student$ = ''                ''
120 LET student$(11 TO 15) = fach$(11 TO 15)
```

Wir haben gesagt, daß GANS ein Teil des String "AQWERGANSLOPU" ist. Der definierende Begriff:

```
(6 to 9)
```

wird als **Teiler** bezeichnet. Er hat noch andere Verwendungszwecke. Beachten Sie, wie dieselbe Schreibweise auf beiden Seiten der **LET**-Anweisung benutzt werden kann. Möchten Sie sich auf ein einzelnes Zeichen beziehen, so wäre die Schreibweise:

```
wirrwarr$(6 to 6)
```

zu umständlich, um nur den Buchstaben "G" (möglicherweise als Schlüssel) herauszunehmen. Statt dessen können Sie:

```
wirrwarr$(6)
```

schreiben, um sich auf ein einzelnes Zeichen zu beziehen.

DATENTYP-UMWANDLUNG

Angenommen, Sie verfügen über eine Variable, *note$*, in der Prüfungsnoten stehen. Nun kann der Teil mit der Note für Geschichte ausgegeben und verbessert werden, weil vielleicht der Geschichtslehrer eine zu strenge Note gegeben hat. Mit folgenden Zeilen wird die Note für Geschichte ausgegeben:

```
100 LET note$ = ''625671''
110 LET gesch$ = note$(3 TO 4)
```

Nun haben wir jedoch das Problem, daß der Wert "56" der Variablen, *gesch$*, aus einer Zeichen-String und nicht aus numerischen Daten besteht. Möchten Sie die Note nun verbessern, indem Sie sie beispielsweise mit 1.125 multiplizieren, so muß der Wert von *gesch$* zuerst in numerische Daten umgewandelt werden. SuperBASIC führt diese Umwandlung automatisch aus, wenn folgende Eingabe vorgenommen wird:

```
120 LET zahl = 1.125 * gesch$
```

Danach wandelt Zeile 120 den String "56" in die Zahl 56 um und multipliziert sie mit 1.125. Dies ergibt einen Wert von 63.

Nun muß die alte Note durch die neue Note ersetzt werden. Die neue Note ist jedoch in der Variablen Zahl gespeichert, also vom Typ numerisch. Bevor sie wieder in den Original-String eingefügt werden kann, muß sie wieder in den String "63" umgewandelt werden. Auch hier wandelt SuperBASIC die Zahl automatisch um, wenn folgende Eingabe vorgenommen wird:

```
130 LET note$(3 TO 4) = zahl
140 PRINT note$
```

Die Ausgabe aus dem ganzen Programm lautet:

```
626371
```

Die Note für Geschichte wurde also auf 63 verbessert.

Genau genommen ist es unzulässig, Datentypen in einer **LET**-Anweisung zu mischen. Es hat keinen Sinn,

```
LET zahl = ''gans''
```

zu schreiben. Sollten Sie dies versuchen, so erhalten Sie eine Fehlermeldung. Schreiben Sie jedoch:

```
LET zahl = ''65''
```

so schließt das System daraus, daß die Zahl 65 zum Wert von Zahl werden soll und führt dies aus. Das vollständige Programm sieht folgendermaßen aus:

```
100 LET note$ = ''625671''
110 LET gesch$ = note$(3 TO 4)
120 LET zahl = 1.125 * gesch$
130 LET note$(3 TO 4) = zahl
140 PRINT note$
```

Auch hier ist die Ausgabe wieder dieselbe!

In Zeile 120 wurde ein String-Wert in numerische Form umgewandelt, so daß er multipliziert werden konnte. In Zeile 130 wurde eine Zahl in einen String umgewandelt. Diese Umwandlung wird als **Datentypumwandlung** bezeichnet.

Nun, nachdem Sie die Begriffe der Teilstrings und der Datentypumwandlung kennen, können Sie das Programm wirtschaftlicher schreiben:

```
100 LET note$ = ''625671''
110 LET note$(3 TO 4) = 1.125 * note$(3 TO 4)
120 PRINT note$
```

Haben Sie schon mit anderen BASIC-Versionen gearbeitet, so werden Sie die Einfachheit und Leistungsfähigkeit der Teilstrings und der Datentypumwandlung schätzen.

DURCHSUCHEN EINES STRING

Sie können einen String nach einem bestimmten Teilstring untersuchen. Mit dem folgenden Programm wird ein Wirrwarr an Buchstaben angezeigt. Sie werden aufgefordert, das Tier einzugeben.

```
100 REMark Tiere suchen
110 LET wirrwarr$ = ''RENTE''
120 PRINT wirrwarr$
130 INPUT ''Wie heißt das Tier ?'' ! tier$
140 IF tier$ INSTR wirrwarr$ AND tier$(1) = ''E''
150  PRINT ''Richtig''
160 ELSE
170  PRINT ''Nicht richtig''
180 END IF
```

Der Operator **INSTR** gibt Null zurück, wenn der gesuchte String nicht im zu durchsuchenden enthalten ist. Ist er enthalten, gibt **INSTR** die Zahl zurück, die der Anfangsposition des Teilstring entspricht, in diesem Fall 6.

Da der Ausdruck:

```
tier$ INSTR wirrwarr$
```

als bedingter Ausdruck behandelt werden kann, kann die Position des String bei einer erfolgreichen Suche als wahr betrachtet werden, während sie bei einer nicht erfolgreichen Suche als *falsch* betrachtet werden kann.

ANDERE STRING-FUNKTIONEN

Sie kennen die **LEN**-Anweisung schon, von der die Länge (Anzahl von Zeichen) eines String zurückgegeben wird.

Möglicherweise möchten Sie einen bestimmten String oder ein bestimmtes Zeichen mehrmals wiederholen. Möchten Sie beispielsweise eine Zeile mit Sternchen ausgeben anstatt 40 Sternchen in einer **PRINT**-Anweisung einzugeben oder eine Schleife zu erstellen, so können Sie einfach folgenden Befehl schreiben:

```
PRINT FÜLLE$ (''*'',40)
```

Schließlich können mit der Funktion **CHR$** interene Codes in String-Zeichen umgewandelt werden. Mit:

```
PRINT CHR$(65)
```

würde beispielsweise A ausgegeben.

VERGLEICHEN VON STRINGS

Bei der Organisation von Daten im Hinblick auf ein schnelles Durchsuchen sind umfangreiche Rechenarbeiten erforderlich. Gelegentlich müssen die Daten in alphabetischer Reihenfolge sortiert werden. Die Grundlage der verschiedenen Sortierverfahren ist der Vergleich zweier Strings, mit denen ermittelt wird, welcher String an erster Stelle steht. Da die Buchstaben A, B, C . . . intern als 65, 66, 67 . . . codiert sind, werden folgende Anweisungen natürlich als richtig betrachtet:

A ist kleiner als B
B ist kleiner als C

Da intern Zeichen für Zeichen verglichen wird, ergibt sich folgendes:

HUND ist kleiner als KATZE
KARTE ist kleiner als KATZE

So können Sie beispielsweise

```
IF ''KATZE'' < ''HUND'' THEN PRINT ''WAU''
```

schreiben. Daraus ergibt sich folgende Ausgabe:

```
WAU
```

Gleichermaßen ergibt:

```
IF ''HUND'' > ''KATZE'' THEN PRINT ''WOOF''
```

folgende Ausgabe:

```
WOOF
```

Für String-Vergleiche werden die Vergleichssymbole der Mathematik benutzt. All die folgenden Bedingungen sind sowohl zulässige Ausdrücke als auch *wahr.*

```
''ALF'' < ''BEN''
''KIT'' > ''BEN''
''KIT'' <= ''LEN''
''KIT'' >= ''KIT''
''PAT'' >= ''LEN''
''LEN'' <= ''LEN''
''PAT'' <> ''PET''
```

Bis jetzt beruhten die Vergleiche ausschließlich auf sinnvollen internen Codes. Die Daten sind jedoch nicht immer so bequem auf Großbuchstaben beschränkt. So möchten wir beispielsweise, daß:

Katze kleiner ist als KIT
und K2N kleiner ist als K27N

Ein einfacher zeichenweiser Vergleich auf der Grundlage interner Codes würde nicht zu diesen Ergebnissen führen. Deshalb verhält sich SuperBASIC etwas intelligenter. Mit dem folgenden Programm, bei dem Ein- und Ausgabe angegeben werden, werden die Regeln für den Vergleich von Strings verdeutlicht.

```
100 REMark Vergleiche
110 REPeat text
120   INPUT "Geben Sie einen String ein" ! erster$
130   INPUT "Geben Sie noch einen String ein" ! zweiter$
140   IF erster$ < zweiter$ THEN PRINT "Kleiner"
150   IF erster$ > zweiter$ THEN PRINT "Größer"
160   IF erster$ = zweiter$ THEN PRINT "Gleich"
170 END REPeat text
```

Eingabe		Ausgabe
KATZE	KIT	Größer
KATZE	KATZE	Gleich
PETE	PETER	Kleiner
K6	K7	Kleiner
K66	K7	Größer
K12N	K6N	Größer

VERGLEICHS-OPERATOREN

> Größer als – Vergleich unter Berücksichtigung von Groß- und Kleinschreibung, Zahlen werden in numerischer Reihenfolge verglichen.

< Kleiner als – Groß- und Kleinschreibung werden berücksichtigt, die Zahlen werden in numerischer Reihenfolge verglichen.

= Gleich – Groß- und Kleinschreibung werden berücksichtigt, Strings müssen identisch sein.

== Fast gleich – Die Strings müssen "annähernd" gleich sein. Groß- und Kleinschreibung werden nicht berücksichtigt, Zahlen werden in numerischer Reihenfolge verglichen.

>= Größer oder gleich – Groß- und Kleinschreibung werden berücksichtigt, Zahlen werden in numerischer Reihenfolge verglichen.

<= Kleiner oder gleich – Groß- und Kleinschreibung werden berücksichtigt, die Zahlen werden in numerischer Reihenfolge verglichen.

AUFGABEN ZU KAPITEL 11

1. Setzen Sie zwölf verschiedene Buchstaben in eine String-Variable und sechs weitere Buchstaben in eine zweite String-Variable. Durchsuchen Sie den ersten String nach jedem der sechs Buchstaben, wobei Sie in jedem Fall angeben, ob der Buchstabe gefunden wurde oder nicht.
2. Wiederholen Sie diese Aufgabe, wobei Sie Tabellen mit einzelnen Zeichen anstelle von Strings benutzen. Setzen Sie 20 zufällige Großbuchstaben in einen String und listen Sie die Buchstaben auf, die wiederholt werden.
3. Schreiben Sie ein Programm, das einen Mustertext in Großbuchstaben liest. Zählen Sie die Häufigkeit jedes Buchstabens und drucken Sie die Ergebnisse aus.

   "FRAU MÜLLER HATTE EINIGE HÜHNER ZU VERKAUFEN, DIE ALLE GESUND UND MUNTER WAREN."
4. Schreiben Sie ein Programm, mit dem die Zahl von Wörtern in dem folgenden Test gezählt wird. Ein Wort wird erkannt, wenn es mit einem Buchstaben beginnt und von einem Leerzeichen, einem Punkt oder einem anderen Satzzeichen abgeschlossen wird.

   "LIEBE EVA, HIERMIT SENDE ICH DIR LIEBE GRÜSSE – DEIN ROLAND .".
5. Schreiben Sie das letzte Programm neu, und benutzen Sie logische Variablen und Prozeduren.

KAPITEL 12
BILDSCHIRM-AUSGABE

Bei SuperBASIC sind die Funktionen für die Bildschirmdarstellung so erweitert worden, daß die verschiedenen Möglichkeiten in zwei Abschnitten beschrieben werden: *Einfaches Drucken* und *Bildschirm.*

Im ersten Abschnitt wird die Ausgabe eines normalen Textes erläutert. Hier werden die bekannten einfachen Methoden zur Anzeige von Meldungen, Texten und numerischen Ausgaben beschrieben. Selbst in diesem einfachen Abschnitt gibt es ein neues Konzpet der 'intelligenten' Platzeinteilung – ein Beispiel für Einfachheit und Nützlichkeit.

Der zweite Abschnitt ist wesentlich umfangreicher, da er viele Punkte umfaßt. Das breite Spektrum an Funktionen gestaltet die Dinge in der Tat einfacher. So können Sie beispielsweise einen Kreis zeichnen, indem Sie einfach das Wort **CIRCLE** schreiben, auf das einige wenige Angaben folgen, mit denen Dinge wie Position und Größe definiert werden.

Bei vielen anderen Systemen müssen Sie Kenntnisse der Geometrie und Trigonometrie haben, um im Grunde genommen einfache Dinge auszuführen.

Jeder Befehl wurde sorgfältig im Hinblick auf seine Funktion ausgewählt. Mit **WINDOW** wird ein Bildschirmbereich definiert. Mit **BORDER** wird eine Umrandung um das Fenster gesetzt. **PAPER** definiert die Hintergrund- oder Papierfarbe, während mit **INK** die Farbe bestimmt wird, mit der auf das Papier geschrieben wird.

Nachdem Sie diesen Abschnitt durchgearbeitet und etwas Erfahrung gewonnen haben, werden Sie sich problemlos an die Funktionen der einzelnen Befehle erinnern. Sie werden diese zusätzlichen Möglichkeiten ohne Schwierigkeiten in Ihre eigene Programmierarbeit einbauen. Mit etwas Erfahrung können Sie sehen, warum die Computer-Grafik zu einer neuen Kunstform wird.

EINFACHES DRUCKEN

Auf den Befehl **PRINT** können eine Reihe von Druckposten folgen. Hier kann es sich um folgende Posten handeln:

Text wie beispielsweise "Dies ist Text"
Variable wie: *num, wort$*
Ausdrücke: 3 * *num, tag$ & woche*

Die Druckposten können in einer Druckanweisung gemischt werden. Zwischen jedem Paar mit Druckposten müssen jedoch eines oder mehrere Trennzeichen stehen. Folgende Trennzeichen können benutzt werden:

- ; Keine Auswirkung – trennt nur die Druckelemente voneinander.
- ! Fügt normalerweise ein Leerzeichen zwischen ausgegebene Werte ein. Paßt ein Wert nicht auf die aktuelle Zeile, so verhält es sich wie ein Symbol für den Zeilenvorschub. Steht ein Wert am Anfang einer Zeile, so wird kein Leerzeichen generiert.
- , Mit einem Tabulator wird die Ausgabe in Spalten von jeweils acht Zeichen angeordnet.
- \ Mit dem Symbol für den Zeilenvorschub wird eine neue Zeile begonnen.
- TO Ermöglicht Tabstops.

Die Zahlen 1, 2, 3 sind zulässige Druckposten und können zur Darstellung der Auswirkungen von Trennzeichen benutzt werden.

Anweisung	Auswirkung
`100 PRINT 1,2,3`	1 2 3
`100 print 1!2!3!`	1 2 3
`100 PRINT 1\2\3`	1 2
`110 PRINT wort$`	Verschiebt die Druckposition
`100 LET zahl = 13` `110 PRINT zahl`	13
`100 LET an$ = ''Ja''` `110 PRINT ''Ich sage'' ! an$`	`Ich sage JA`
`110 PRINT ''Sum ist'' ! 4 + 2`	`Sum is 6`

Sie können die Druckausgabe mit dem **AT**-Befehl an eine beliebige Stelle auf dem Bildschirm setzen.

Zum Beispiel:

```
AT 10,15 : PRINT ''Das ist in Zeile 10 Spalte 15''
```

Mit dem **CURSOR**-Befehl kann die Druckausgabe an eine beliebige Stelle in dem Pixel-Koordinatensystem des Bildschirms gesetzt werden.

Zum Beispiel:

```
CURSOR 100,150 : PRINT ''Das ist 100 Pixel-Raster Einheiten
waagerecht und 150 senkrecht''
```

Bei der Lektüre des Kapitels *Befehle* werden Sie Schwierigkeiten haben, den Abschnitt über **PRINT** mit der obigen Beschreibung in Einklang zu bringen. Zwei der Schwierigkeiten sind gelöst, sobald Sie folgendes verstanden haben:

Text in Anführungszeichen, Variablen und Zahlen sind genau genommen Ausdrücke; sie stellen die einfachste Form der Ausdrücke dar.

Druck-Trennzeichen werden einfach als Druckposten eingeordnet.

BILDSCHIRM

In diesem Abschnitt werden allgemeine Gesichtspunkte dargelegt, die für die Ausgabe von Text oder Grafiken erforderlich sind. Mit der Anweisung:

`MODE 8` oder `MODE 256`

wird **MODE 8** ausgewählt. In dieser Betriebsart gelten folgende Angaben:

- 256 Pixel waagrecht mit den Nummern 0 bis 511 (zwei Nummern pro Pixel)
- 256 Pixel senkrecht mit der Nummer 0 bis 255
- 8 Farben

Ein Pixel ist der kleinste Farbbereich, der angezeigt werden kann. Wir benutzen den Ausdruck, **Vollfarbe**, da sie mit normal aussehenden Farben verglichen werden können, von denen es nur acht gibt. Mit Hilfe einer Reihe von Effekten können eine Vielzahl von Schattierungen und Strukturen erstellt werden. Benutzen Sie den QL mit einem normalen Fernsehgerät, so können diese zusätzlichen Effekte auf dem Bildschirm nicht wiedergegeben werden.

Mit der Anweisung:

`MODE 4` oder `MODE 512`

wird **MODE 4** ausgewählt. In dieser Betriebsart gelten folgende Angaben:

- 512 Pixel waagrecht, von 0 bis 511 numeriert
- 256 Pixel senkrecht, von 0 bis 255 numeriert
- 4 Farben

FARBE

Mit dem folgenden Code zusammen mit den entsprechenden Befehlen wie **PAPER**, **INK** usw. können Sie eine Farbe auswählen. Beachten Sie, daß die Zahlen selbst keine Bedeutung haben. Die Zahlen werden nur als Farben interpretiert, wenn sie mit **PAPER** und **INK** usw. benutzt werden.

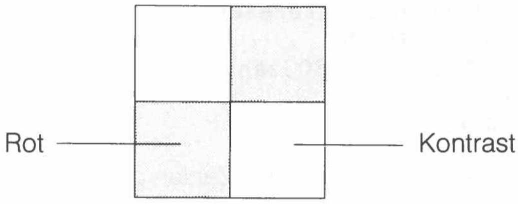
MODE 8	Code	MODE 4
Schwarz	0	Schwarz
Blau	1	Schwarz
Rot	2	Rot
Magenta	3	Rot
Grün	4	Grün
Cyan	5	Grün
Gelb	6	Weiß
Weiß	7	Weiß

Farbcode

So ergibt beispielsweise **INK** 3 Magenta im **MODUS** 8.

PUNKTMUSTER

Sie können auch zwei Farben in einer entsprechenden Anweisung angeben. So würde 2,4 beispielsweise ein Punktemuster wie nachfolgend dargestellt ergeben. In jeder Gruppe mit vier Pixeln Rot (Code 2) je nach der zuerst ausgewählten Farbe. Die beiden anderen Pixel wären ein Kontrast. Dieser Effekt kann auf einem normalen Fernsehbildschirm nicht wiedergegeben werden.



Schreiben Sie:

```
INK 2,4
```

so wird eine Mischfarbe aus den beiden Codes 2 und 4 gebildet. Diese Codes werden mit Farbe und Kontrast bezeichnet!

```
INK Farbe, Kontrast
```

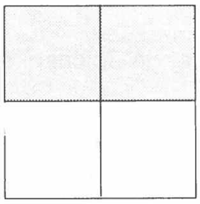
Sie können die Auswirkungen der Punktmuster selbst sehen, indem Sie sie ausprobieren. Hier wollen wir jedoch noch einige technische Details angeben.

```
100 REMark Farbe / Kontrast
110 FOR farbe = 0 TO 7 STEP 2
120   PAPER farbe: CLS
130   FOR kontrast = 0 TO 7 STEP 2
140     BLOCK 100,50,40,50,farbe,kontrast
150     PAUSE 50
160   END FOR kontrast
170 END FOR farbe
```

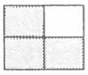
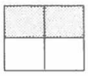
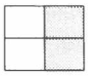
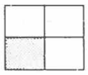
Möchten Sie mit verschiedenen Punktmustern experimentieren, so können Sie zu der Farbangabe eine dritte Codenummer hinzufügen. Zum Beispiel würde:

```
INK 2, 4, 1
```

ein rot/grünes waagerechtes Streifenmuster ergeben. Ein Block mit vier Pixeln würde folgendermaßen aussehen:



Die möglichen Effekte werden mit Rot ☐ und Kontrast ☐ dargestellt.

Code	Name	Effekt
0	Einzelnes Kontrast-Pixel	
1	Waagerechte Streifen	
2	Senkrechte Streifen	
3	Schachbrettmuster	

Punktmuster

FARBPARAMETER

Sie können einen Farb-/Punktmuster-Effekt wie oben beschrieben mit drei Zahlen erzeugen. So könnte beispielsweise:

INK Farbe, Kontrast, Punkmuster

mit folgenden Werten benutzt werden:

Farbe in Bereich 0 bis 7
Kontrast in Bereich 0 bis 7
Punktmuster in Bereich 0 bis 3

Sie können denselben Effekt mit einer einzigen Zahl erzielen, auch wenn dies nicht ganz so einfach ist. Hier wird auf den Abschnitt *Begriffe – Farbe* verwiesen.

Mit dem folgenden Programm werden alle möglichen Farbeffekte angezeigt:

```
100 REMark Farben-Spielereien
110 FOR zahl = 0 TO 255
120   BLOCK 100,50,40,50,zahl
130   PAUSE 50
140 END FOR zahl
```

PAPER

PAPER gefolgt von einer, zwei oder drei Zahlen gibt die Hintergrundfarbe an. Zum Beispiel:

`PAPER 2`	{Rot}
`PAPER 2,4`	{Rot/Grünes Schachbrettmuster}
`PAPER 2,4,1`	{Rot/Grüne horizontale Streifen}

Die Farbe wird erst sichtbar, wenn etwas anderes geschieht, beispielsweise der Bildschirm durch Eingabe von **CLS** gelöscht wird.

INK

Mit **INK**, gefolgt von einer, zwei oder drei Zahlen, wird der Code für das Ausdrucken von Zeichen, Linien oder anderen Grafiken angegeben. Die Farb- und Punktmuster-Effekte sind dieselben wie bei **PAPER**. Zum Beispiel:

`INK 2`	{Rote Schriftarbe}
`INK 2,4`	{Rot/Grünes Punktmuster als Schriftfarbe}
`INK 2,4,1`	{Rot/Grünes waagerechtes Streifenmuster als Schriftfarbe}

Die Schriftfarbe gilt für die nachfolgende Ausgabe bis sie erneut geändert wird.

CLS

Mit **CLS** wird das Fenster auf die aktuelle Papierfarbe gelöscht – wie ein Lehrer, der die Tafel löscht, nur daß es sich hier um ein elektronisches und mehrfarbiges Fenster handelt.

BLINKEN

Die Schriftfarbe kann nur bei **MODE 8** blinken. Um das Blinken einzuschalten, wird:

```
FLASH 1
```

eingegeben. Um es wieder auszuschalten, wird:

```
FLASH 0
```

eingegeben. Überlagern sich blinkende Zeichen, so kann dies zu alarmierenden Ergebnissen führen.

DATEIEN

Sie haben schon Microdrives für die Speicherung von Programmen mit den Befehlen **LOAD** und **SAVE** benutzt. Kassetten können für die Speicherung von Daten und Programmen benutzt werden. Mit dem Wort Datei wird im allgemeinen eine Folge von Datensätzen bezeichnet, wobei ein Datensatz eine Gruppe von zusammengehörigen Informationen, wie beispielsweise Namen, Adresse und Telefonnummer darstellt.

Zwei der gebräuchlichsten Dateitypen sind serielle Dateien und Dateien für den direkten Zugriff. Die Daten in einer seriellen Datei werden im allgemeinen in ihrer Reihenfolge ab dem ersten Eintrag gelesen. Soll der 50. Eintrag gelesen werden, so müssen zuerst die ersten 49 Einträge gelesen werden, um diesen Eintrag zu finden. Dagegen kann der 50. Eintrag in einer Datei für den direkten Zugriff schnell gefunden werden, da sich das System nicht durch die früheren Datensätze durcharbeiten muß, um zu diesem Datensatz zu gelangen. Popmusik auf einer Kassette kann mit einer seriellen Datei verglichen werden. Acht Stücke auf einer Langspielplatte können dagegen mit einer Datei für den direkten Zugriff verglichen werden. Sie können den Tonarm direkt auf eine der acht Spuren setzen.

Der einfachste Dateityp besteht einfach aus einer Folge von Zahlen. Um diese Idee darzustellen, setzen wir die Zahlen 1 bis 100 in eine Datei namens *Zahlen*. Der vollständige Dateiname besteht jedoch aus zwei Teilen:

- Einheitenname
- angehängte Informationen

Angenommen, wir wollen die Datei, *Zahlen*, auf einer Kassette in Microdrive 1 erstellen. Der Einheitenname lautet:

```
mdv1_
```

Die angehängte Information besteht einfach aus dem Namen der Datei:

```
zahlen
```

So lautet der vollständige Dateiname:

```
mdv1_zahlen
```

KANÄLE

Ein Programm kann mehrere Dateien gleichzeitig benutzen. Die Bezugnahme auf eine Datei geschieht jedoch bequemer durch eine zugeordnete Kanalnummer. Hier kann es sich um eine ganze Zahl in dem Bereich von 0 bis 15 handeln. Eine Datei wird mit der **OPEN**-Anweisung oder, wenn es sich um eine neue Datei handelt, mit der **OPEN_NEW**-Anweisung mit einer Kanalnummer verknüpft. So könnten Sie beispielsweise Kanal 7 für die Datei "Zahlen" auswählen und:

```
OPEN_NEW #7,mdv1_zahlen
|        |  |    |
|        |  |    └──── Datei
|        |  └───────── Einheit
|        └──────────── Kanalnummer
└───────────────────── Befehl
```

schreiben.

Nun können Sie sich nur durch Angabe der Nummer #7 auf die Datei beziehen. Das vollständige Programm sieht folgendermaßen aus:

```
100 REMark Einfache Datei
110 OPEN_NEW #7, mdv1_zahlen
120 FOR nummer = 1 TO 100
130   PRINT #7 , nummer
140 END FOR nummer
150 CLOSE #7
```

Mit der **PRINT**-Anweisung werden die Zahlen auf der Kassettendatei "ausgedruckt", da ihr die Nummer #7 zugewiesen wurde. Die Anweisung **CLOSE #7** ist erforderlich, da das System einige interne Arbeiten ausführen muß, nachdem die Datei benutzt wurde. Mit ihr wird auch Kanal 7 für andere Zwecke freigestellt. Nachdem das Programm ausgeführt wurde, geben Sie:

```
DIR mdv1_
```

ein. In dem Inhaltsverzeichnis können Sie sehen, daß die Datei *Zahlen* auf der Kassette in Microdrive mdv1_ steht.

Nun müssen Sie jedoch auch wissen, ob die Datei korrekt aufgezeichnet wurde. Dies können Sie nur, wenn die Datei gelesen und überprüft wird. Hierzu ist der Befehl **OPEN_IN** erforderlich. Ansonsten ähnelt das Programm für das Auslesen von Daten aus einer Datei dem vorhergehenden Programm.

```
100 REMark Datei-Lesen
110 OPEN_IN #6, mdv1_zahlen
120 FOR posten = 1 TO 100
130   INPUT #6, nummer
140   PRINT ! nummer !
150 END FOR posten
160 CLOSE #6
```

Das Programm muß die Zahlen 1 bis 100 ausgeben, jedoch nur wenn die Kassette mit der Datei *Zahlen* noch in Microdrive mdv1_ eingelegt ist.

EINHEITEN UND KANÄLE

Sie haben ein Beispiel für eine Einheit gesehen, eine Datendatei in einem Microdrive. Hier kann davon gesprochen werden, daß eine Datei geöffnet wurde. Genau genommen meinen wir jedoch, daß eine Einheit mit einem bestimmten Kanal verknüpft wurde. Außerdem wurden die weiter erforderlichen Informationen angegeben. Einige Einheiten sind vom System permanent Kanäle zugewiesen:

Kanal	Benutzung (standardmäßig)
#0	AUSGABE: Befehlsfenster EINGABE: Tastatur
#1	EINGABE: Druck-Fenster
#2	AUSGABE: Programmauflistung

FENSTER

Sie können ein Fenster mit einer beliebigen Größe an einer beliebigen Stelle auf dem Bildschirm erstellen. Der Einheitenname für ein Fenster lautet:

```
scr
```

An diesen Namen wird beispielsweise folgende Information angehängt:

```
scr_360 x 50a80 x 40
```

- Y-Koordinate (Ecke links oben)
- X-Koordinate (Ecke links oben
- Höhe
- Breite

Mit dem folgenden Programm wird ein Fenster mit der Kanalnummer 5 erstellt und mit grüner Farbe (Code 4) ausgefüllt. Danach wird es geschlossen:

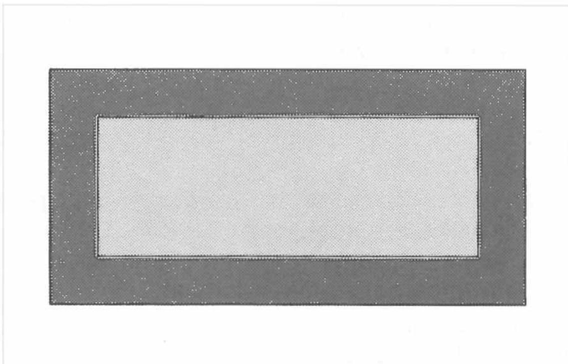
```
100 REMark Fenster definieren
110 OPEN #5, scr_400 x 200a20 x 50
120 PAPER #5,4 : CLS #5
130 CLOSE #5
```

Beachten Sie, daß jedes Fenster über eigene Angaben, wie beispielsweise Hintergrundfarbe, Schriftfarbe usw., verfügen kann. Die Tatsache, daß ein Fenster geöffnet wurde, bedeutet nicht, daß es sich um das aktuelle Standardfenster handelt.

Sie können die Position oder Form eines eröffneten Fensters ändern, ohne dieses zu schließen und wieder öffnen zu müssen. Versuchen Sie, zwei Zeilen zu dem vorigen Programm hinzuzufügen:

```
124 WINDOW #5,300,100,110,65
126 PAPER #5,2 : CLS #5
```

Führen Sie das Programm erneut aus und Sie werden ein rotes Fenster innerhalb des ursprünglich grünen Fensters finden. Dieses rote Fenster ist nun mit Kanal 5 verknüpft, siehe folgende Abbildung.



UMRANDUNG

Sie können um den Rand des Bildschirms oder eines Fensters eine Umrandung setzen. Mit:

```
BORDER #5,6
```

wird beispielsweise ein Umrandung um das mit Kanal #5 verknüpfte Fenster gezeichnet. Diese Umrandung hat eine Stärke von sechs Pixeln und die Größe des Fensters wird dementsprechend gekürzt. Die Umrandung ist transparent, wobei darunterstehende Angaben geschützt werden. Sie können mit der üblichen Methode auch eine farbige Umrandung angeben.

```
BORDER #5,6,2
```

Erzeugt eine rote Umrandung. Mit den üblichen Methoden können Sie auch Umrandungen mit anderen Farben und Strukturen zeichnen. Zum Beispiel:

```
BORDER 10
```

fügt eine transparente Umrandung mit einer Stärke von 10 Pixeln zu dem aktuellen Fenster hinzu. (Die Umrandung ist transparent, da keine Farbe angegeben wurde.) Mit:

```
BORDER 2,0,7,0
```

wird eine Umrandung in einem schwarz/weißen Punktmuster mit einer Stärke von 2 Pixeln hinzugefügt.

BLOCK

Sie können die Größe, Position und Farbe eines Blockes mit einer einzigen Anweisung angeben. Sie wird in das Pixel-Koordinatensystem bezogen auf das aktuelle Fenster oder den aktuellen Bildschirm gestellt. Mit:

```
BLOCK #5 ,10,20,50,100,2
```

wird beispielsweise ein Block in dem Fenster #5 bei Position 50 waagerecht und 100 senkrecht erstellt. Der Block hat eine Breite von 10 Einheiten und eine Höhe von 20 Einheiten. Seine Farbe ist Rot.

Hier wird darauf hingewiesen, daß die **WINDOW**- und **BLOCK**-Anweisungen ohne Änderung im 4- und 8-Farb-Modus arbeiten (auch wenn die Farben unterschiedlich sein können). Dies ist darauf zurückzuführen, daß die waagrechte Achse stets in 512 (Pixel-)Einheiten unterteilt ist und daß senkrecht stets 256 Pixelpositionen vorhanden sind.

BESONDERE DRUCK-VERFAHREN CSIZE

Sie können die Größe der Zeichen ändern. Mit:

```
CSIZE 3,1
```

erhalten Sie beispielsweise die größtmöglichen Zeichen, während Sie mit:

```
CSIZE 0,0
```

die kleinsten Zeichen erhalten. Die erste Zahl muß 0, 1, 2 oder 3 sein. Mit ihr wird die Breite bestimmt.

Die zweite Zahl muß 0 oder 1 sein und legt die Höhe fest. Die normalen Größen sind:

```
MODE 4 CSIZE 0,0
MODE 8 CSIZE 2,0
```

Die Anzahl von Zeilen und Spalten für jede Zeichengröße hängt davon ab, ob die Ausgabe auf einem Monitor oder einem Fernsehgerät angezeigt wird. Hier werden die Zeilen- und Spalten-Größen für einen Monitor angegeben. Die Größen bei einem Fernsehgerät sind kleiner und variieren auch für die einzelnen Fernsehempfänger.

In der niedrigauflösenden Betriebsart können Sie bei dem QL keine kleinere Zeichengröße als die Standardgröße auswählen.

STREIFEN

Normalerweise wird auf der aktuellen Papierfarbe gedruckt. Diese können Sie mit einem Streifenmuster ändern.

Sie können einen speziellen Hintergrund für Zeichen wählen, mit dem diese hervorgehoben werden können. Durch:

```
STRIP 7
```

erhalten Sie beispielsweise einen weißen Streifen, während Sie mit:

```
STRIP 2,4,2
```

einen Streifen erhalten, der selbst ein senkrechtes, rot/grünes Streifenmuster besitzt. Alle normalen Farbkombinationen sind möglich.

ÜBERSCHREIBEN

Mit den folgenden Befehlen können Sie weitere Effekte erzielen:

`OVER 1`	Überschreibt bestehenden Text ohne etwas zu löschen
`OVER -1`	Bildet bei jedem Pixel das exclusive Oder (XOR) mit dem vorhandenen und dem neu zu druckenden Pixel. Ist das Ergebnis wahr, (d. h. liegt einmal Schriftfarbe und das andere mal Hintergrundfarbe vor) dann wird das Pixel in Schriftfarbe, andernfalls in Hintergrundfarbe gesetzt.

Um zu dem normalen Ausdrucken auf dem aktuellen Streifen zurückzukehren, benutzen Sie:

```
OVER 0
```

UNTERSTREICHEN

Zeichen können unterstrichen werden.

`UNDER 1`	Unterstreicht die nachfolgende Ausgabe in der aktuellen Schriftfarbe.
`UNDER 0`	Schaltet das Unterstreichen aus.

GRAFIKEN

Da der QL, je nach Betriebsart verschieden geformte Pixel ausgibt, könnten bei der Grafik Probleme entstehen. Was in der einen Betriebsart wie ein Quadrat aussieht würde in der anderen als Rechteck erscheinen. Deshalb benutzen die Grafikprozeduren von SuperBASIC ein eigenes Koordinatensystem, das unabhängig von der Betriebsart ist.

Standardmäßig ist beim grafischen Koordinatensystem die senkrechte y-Achse eines Fensters immer in 100 Einheiten unterteilt. Die Einheit auf der waagerechten x-Achse ist gleich lang wie die Einheit auf der y-Achse. Der Ursprung liegt standardmäßig in der linken unteren Ecke des jeweiligen Fensters.

Der Maßstabsfaktor, d.h. die Anzahl von Einheiten, in die die y-Achse unterteilt ist, sowie die Lage des Koordinatenursprungs können mit dem **SCALE**-Befehl verändert werden.
Ein Grafik-Cursor wird vom System automatisch gesetzt. Er steht jeweils auf dem zuletzt gezeichneten Punkt.

PUNKTE UND LINIEN

Punkte und Linien können problemlos gezeichnet werden. Bei einem vertikalen Maßstab von 100 kann ein Punkt in der Nähe der Fenstermitte mit folgendem Befehl gezeichnet werden:

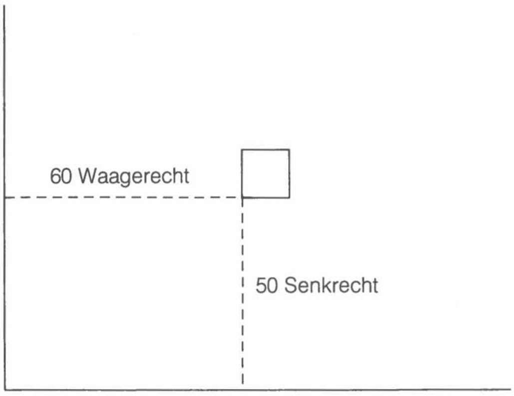
```
POINT 60,50
```

Der Punkt (60 Einheiten waagerecht und 50 Einheiten senkrecht) wird in der aktuellen Schriftfarbe gezeichnet.
Auf ähnliche Weise wird mit folgender Anweisung eine Linie gezeichnet:

```
LINE 60,50 TO 80,90
```

Weitere Elemente können hinzugefügt werden. Mit folgendem Befehl wird beispielsweise ein Quadrat gezeichnet:

```
LINE 60,50 TO 70,50 TO 70,60 TO 60,60 TO 60,50
```



RELATIVER MODUS

Ein Koordinatenpaar wie:

waagerecht, senkrecht

definiert normalerweise einen Punkt bezogen auf den Ursprung 0,0 des Koordinatensystems. Gelegentlich ist es bequemer, Punkte bezogen auf die aktuelle Cursor-Position zu definieren. So kann beispielsweise das obige Quadrat auf andere Art und Weise mit der Anweisung **LINE_R** gezeichnet werden. Sie bedeutet:

"Der Koordinatenursprung wird vorübergehend auf die jeweils letzte Position des Grafik-Cursors gesetzt."

```
POINT 60,50
LINE_R 0,0 TO 10,0 TO 0,10 TO -10,0 TO 0,-10
```

Der erste Punkt 60,50 wird zum Ursprung. Während die Linien gezeichnet werden, wird das Ende einer Linie zum Ursprung für die nächste Linie.
Mit dem folgenden Programm wird ein Muster mit zufällig gesetzten farbigen Quadraten gezeichnet.

```
100 REMark Farbige Quadrate
110 PAPER 7 : CLS
120 FOR quad = 1 TO 100
130   INK RND(1 TO 6)
140   POINT RND(90),RND(90)
150   LINE_R 0,0 TO 10,0 TO 0,10 TO -10,0 TO 0, -10
160 END FOR quad
```

Dasselbe Ergebnis könnte mit der absoluten Grafik erzielt werden, wäre jedoch etwas aufwendiger.

KREISE UND ELLIPSEN

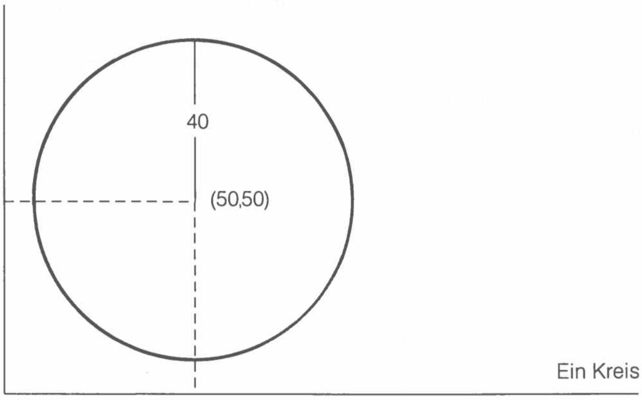
Soll ein Kreis gezeichnet werden, so müssen Sie folgende Angaben machen:

Koordinaten des Mittelpunktes, z.B. 50,50
Radius, z.B. 40

Mit der Anweisung

```
CIRCLE 50,50,40
```

wird ein Kreis mit dem Mittelpunkt bei 50,50 und mit einem Radius (bzw. einer Höhe) von 40 Einheiten gezeichnet. Siehe folgende Abbildung:

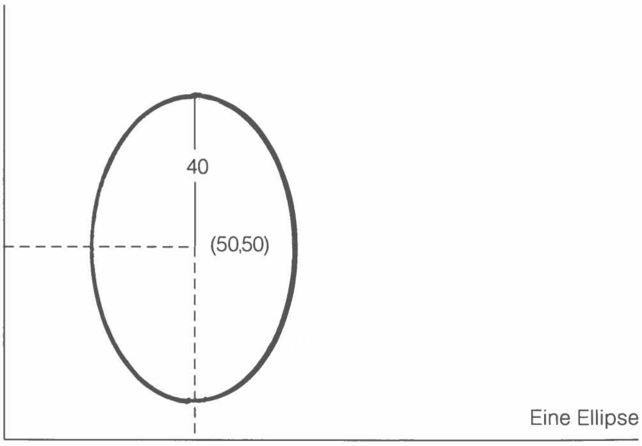


Ein Kreis

Werden zwei weitere Parameter hinzugefügt, z. B.:

```
CIRCLE 50,50,40,.5,0
```

so erhalten Sie eine Ellipse. Die Befehle **CIRCLE** und **ELLIPSE** können beliebig ausgetauscht werden.

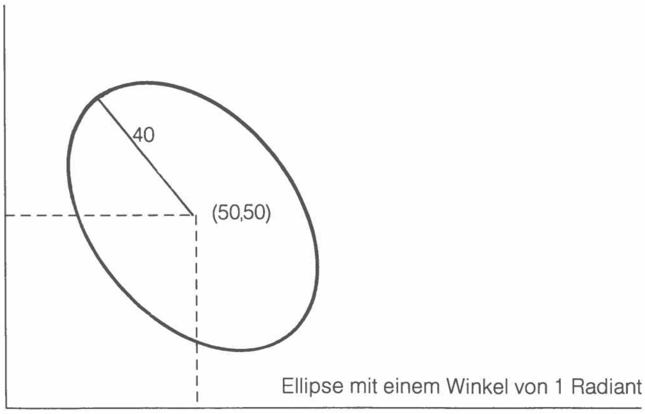


Eine Ellipse

Eine Halbachse der Ellipse beträgt wie vorher 40. Die andere Halbachse beträgt jedoch nur 0.5 der ersten. Die Zahl 0.5 wird als Exzentrizität bezeichnet. Ist die Exzentrizität gleich 1, so erhalten Sie einen Kreis, ist sie jedoch von 1 verschieden und größer als 0, so erhalten Sie eine Ellipse. Soll eine Ellipse geneigt werden, so können Sie den fünften Parameter ändern, z. B.:

```
CIRCLE 50,50,40,.5,1
```

Dadurch wird die Ellipse entgegen dem Uhrzeigersinn um ein Radiant, d. h. etwa 57 Grad, wie in der Abbildung dargestellt, geneigt.



Ellipse mit einem Winkel von 1 Radiant

Ein gestreckter Winkel beläuft sich auf 180 Grad oder **PI** Radiant. Mit folgendem Programm können Sie also ein Muster mit Ellipsen zeichnen:

```
100 FOR drehung = 0 TO 2 * PI STEP PI/6
110   CIRCLE 50,50,40,.5,drehung
120 END FOR drehung
```

Für einen Kreis oder eine Ellipse werden die Parameter in folgender Reihenfolge angegeben:

Mitte_waagerecht, Mitte_senkrecht, Halbachse, [Exzentrizität, Winkel]

Die beiden letzten Parameter können entfallen. Dies wird dadurch angegeben, daß sie in eckigen Klammern stehen ([ ]).

Beispiel Schreiben Sie ein Programm, das folgende Funktionen ausführt:

1. Öffnen eines Fensters von 100 x 100 bei (100,50).
2. Maßstabsfaktor 100 in **MODE** 8.
3. Eine schwarze Hintergrundfarbe wird ausgewählt und der Bildschirm wird gelöscht.
4. Eine grüne Umrandung mit einer Breite von zwei Einheiten wird erstellt.
5. Ein Muster aus sechs farbigen Kreisen wird gezeichnet.
6. Das Fenster wird geschlossen.

```
100 REMark Muster
110 MODE 8
120 OPEN #7,scr_100 x 100a100 x 50
130 SCALE #7,100,0,0
140 PAPER #7,0 : CLS #7
150 BORDER #7,2,4
160 FOR farbe = 1 TO 6
170   INK #7 , farbe
180   LET drehung = 2* PI/farbe
190   CIRCLE #7,50,50,30,.5,drehung
200 END FOR farbe
210 CLOSE #7
```

Durch Änderung des Programms können einige interessante Effekte erzielt werden. Nehmen Sie beispielsweise folgende Änderungen vor:

```
160 FOR farbe = 1 to 100
180 Let drehung = farbe * PI /50
```

BÖGEN

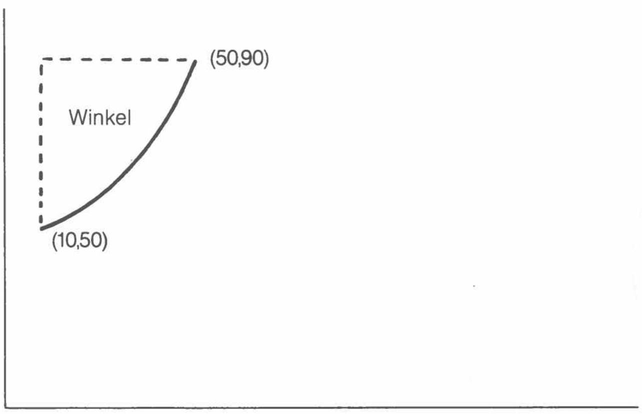
Soll ein Bogen gezeichnet werden, so müssen folgende Entscheidungen gefällt werden:

- Startpunkt
- Endpunkt
- Krümmung.

Die beiden ersten Elemente sind problemlos. Bei der Krümmung verhält sich dies schon etwas anders. Sie können den Bogen zeichnen, indem Sie genau zeichnen oder Versuche unternehmen. Sie müssen jedoch entscheiden, welchen Winkel der Bogen haben soll und den Winkel dann in Radiant angeben. Ein Winkel von 1,5 Radiant ergibt eine scharfe Krümmung, während ein kleiner Winkel eine sehr leichte Krümmung ergibt. Geben Sie beispielsweise folgenden Befehl ein:

```
ARC 10,50 TO 50,90, 1
```

Dadurch erhalten Sie eine leichte Krümmung.



AUSFÜLLEN

Sie können eine geschlossene Figur mit der aktuellen **INK**-Farbe ausfüllen, indem Sie einfach:

```
FILL 1
```

schreiben, bevor die Figur gezeichnet wird. Mit dem folgenden Programm wird ein grüner Kreis erzeugt:

```
INK 4
FILL 1
CIRCLE 50,50,30
```

Bei dem **FILL**-Befehl werden sich berührende waagrechte Linien zwischen entsprechenden Punkten gezeichnet. Vor jeder neuen Figur, die gefüllt werden soll, muß der **FILL**-Befehl gegeben werden.

Mit der Anweisung:

```
FILL 0
```

wird das Ausfüllen wieder ausgeschaltet.

ROLLEN UND WAAGERECHT VERSCHIEBEN

Sie können die Anzeige in einem Fenster wie ein Kameramann in einem Film rollen oder verschieben. Sie geben an, um wieviele Pixel aufwärts oder abwärts gerollt werden soll. Bei einer positiven Anzahl von Pixeln wird nach unten gerollt. So verschiebt:

```
SCROLL 10
```

die Bildschirmanzeige in dem aktuellen Fenster oder Bildschirm um 10 Pixel nach unten.

```
SCROLL -8
```

Verschiebt die Anzeige um 8 Pixel nach unten. Ein teilweises Rollen kann mit einem zweiten Parameter angegeben werden.

```
SCROLL -8,1
```

Rollt den Teil über der Cursor-Zeile (diese nicht eingeschlossen). (Gemeint ist der Cursor, auf den sich z. B. PRINT bezieht.)

```
SCROLL -8,2
```

Rollt den Teil unter der Cursor-Zeile (diese nicht eingeschlossen).

Während des Rollens wird der durch die Verschiebung der Anzeige frei gewordene Platz mit der aktuellen Papierfarbe ausgefüllt. Ein zweiter Parameter Null hat keine Wirkung.

Die Anzeige in dem aktuellen Fenster kann mit **PAN** nach links oder rechts verschoben werden. Die **PAN**-Anweisung wird wie **SCROLL** benutzt, allerdings verschiebt

`PAN 40` die Anzeige nach rechts
`PAN -40` die Anzeige nach links.

Mit einem zweiten Parameter wird ein teilweises Verschieben nach rechts oder links angegeben.

0 – Ganzer Bildschirm
3 – Ganze Zeile, in der der Cursor steht.
4 – Rechter Teil der Zeile, in der der Cursor steht, einschließlich Cursorbereich.

Werden Punktmuster benutzt oder wird im 8-Farb-Modus gearbeitet, so müssen Fenster in Vielfachen von zwei Pixeln verschoben oder gerollt werden.

AUFGABEN ZU KAPITEL 12

1. Schreiben Sie ein Programm, mit dem ein Raster mit 10 Zeilen zu 10 Quadraten gezeichnet wird.
2. Setzen Sie die Zahlen 1 bis 100 in die Quadrate, wobei Sie unten links beginnen und E für Ende in das letzte Quadrat setzen.
3. Zeichnen Sie ein Wurfbrett auf dem Bildschirm. Es soll aus einem äußeren Ring mit Zahlen bestehen. Danach wird ein zweiter und dritter Ring gezeichnet, wobei die Mitte aus einem Bullauge und einem Ring besteht.

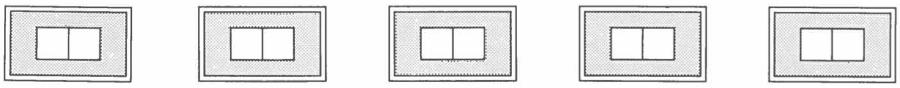
KAPITEL 13
TABELLEN

Angenommen, Sie sind ein Gefängnisdirektor und haben einen neuen Zellenblock, der als Westblock bezeichnet wird. In ihn können 50 neue Häftlinge aufgenommen werden. Nun müssen Sie wissen, welcher Häftling (anhand seiner Nummer) in welcher Zelle untergebracht ist. Sie könnten jeder Zelle einen Namen geben. Es ist jedoch einfacher, ihnen Nummern von 1 bis 50 zuzuweisen.

Hier wollen wir uns nur fünf Häftlinge mit Nummern vorstellen, die wir in eine **DATA**-Anweisung setzen können:

```
DATA 50, 37, 86, 41, 32
```

Wir erstellen eine Tabelle mit Variablen, die alle den Namen West haben und durch eine in Klammern angegebene Nummer voneinander unterschieden werden.



Hier muß eine Tabelle definiert und mit einer **DIM**-Anweisung dimensioniert werden:

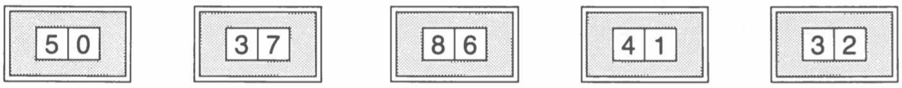
```
DIM west(5)
```

Dadurch kann SuperBASIC den benötigten Speicherplatz zuweisen, auch wenn der Bedarf viel größer ist als in diesem Beispiel. Nachdem die **DIM**-Anweisung ausgeführt wurde, können die fünf Variablen benutzt werden.

Die Häftlinge können mit **READ** aus der **DATA**-Anweisung in fünf Tabellenvariablen gelesen werden:

```
For element = 1 TO 5 : READ west(zelle)
```

Nun fügen wir eine weitere **FOR**-Schleife hinzu. Darin wird mit **PRINT** angezeigt, daß die "Zellen" auch mit den "Häftlingen" richtig besetzt sind.



Das vollständige Programm wird nachfolgend dargestellt:

```
100 REMark Gefängnisverwaltung
110 DIM west(5)
120 FOR zelle = 1 TO 5: READ west(zelle)
130 FOR zelle = 1 TO 5 : PRINT zelle ! west(zelle)
140 DATA 50,37,86,41,32
```

Das Programm gibt aus:

```
1  50
2  37
3  86
4  41
5  32
```

Die Zahlen 1 bis 5 werden als *Indizes* des Tabellennamens *West* bezeichnet. Die Tabelle, *West*, ist eine numerische Tabelle, die aus fünf numerischen Tabellenelementen besteht.

Zeile 130 kann durch:

```
130 PRINT west
```

ersetzt werden. Dadurch werden nur die folgenden Werte ausgegeben:

```
 0
50
37
86
41
32
```

Die Null am Anfang der Liste wird angezeigt, da die Indizes von Null bis zu der, in der **DIM**-Anweisung eingegebenen Grenze geht. Wir werden später noch darlegen, wie nützlich Tabellenelemente mit dem Index 0 sein können.

Beachten Sie bitte, daß den Elementen einer numerischen Tabelle beim Dimensionieren mit **DIM** der Wert Null zugewiesen wird.

STRING-TABELLEN

String-Tabellen sind gleichbedeutend mit numerischen Tabellen. Allerdings gibt eine zusätzliche Dimension in der **DIM**-Anweisung die Länge jeder String-Variablen in der Tabelle an. Angenommen, die zehn besten Golfspieler eines Golfclubs werden mit ihren Vornamen in **DATA**-Anweisungen angegeben.

```
DATA ''Bert'', ''Jochen'', ''Walter'', ''Werner'', ''Dirk''
DATA ''Tom'', ''Karl'', ''Harald'', ''Jan'', ''Achim''
```

Nun benötigen Sie zehn verschiedene Variablennamen. Wären jedoch 100 oder 1000 Spieler vorhanden, so würde die Aufgabe recht mühsam. Eine Tabelle besteht aus einer Gruppe von Variablen, mit denen derartige Probleme gelöst werden sollen. Jeder Name einer Tabellen-Variablen besteht aus zwei Teilen:

- einem nach den üblichen Regeln gebildeten Namen
- einen numerischer Teil, der als Index bezeichnet wird.

Schreiben Sie die Variablennamen als:

`wohnung$(1), wohnung$(2), wohnung$(3)` usw.

Bevor Sie die Tabellen-Variablen benutzen können, müssen Sie dem System Angaben über die Tabelle und ihre Dimensionen liefern:

```
DIM wohnung$(10,7)
```

Mit diesem Befehl werden elf (0 bis 10) Variablen zur Benutzung in dem Programm reserviert. Jede String-Variable in der Tabelle kann maximal acht Zeichen umfassen. Die **DIM**-Anweisungen sollten normalerweise alle zusammen am Anfang des Programms stehen. Nachdem die Tabelle in einer **DIM**-Anweisung definiert wurde, können sämtliche Elemente der Tabelle benutzt werden. Ein wichtiger Vorteil besteht darin, daß Sie den numerischen Teil (den Index) als numerische Variable angeben können. So können Sie:

```
FOR zahl = 1 TO 10 : READ wohnung$(zahl)
```

schreiben. Dadurch werden den Golfern Wohnungen zugewiesen:

wohnung$(1)	wohnung$(2)	wohnung$(3)	-----------	wohnung$(10)
Bert	Jochen	Walter		Achim

Sie können auf die übliche Art und Weise Bezug auf die Variablen nehmen. Denken Sie jedoch daran, den richtigen Index zu benutzen. Angenommen, Bert und Walter wollen ihre Wohnungen tauschen. In der Computersprache ausgedrückt, müßte einer von ihnen, sagen wir Bert, einen Zwischenspeicher (als Wohnung) benutzen, damit Walter umziehen kann. Sie könnten folgende Befehle schreiben:

```
LET momentan$=wohnung$(1) : REMark Tom vorübergehend in momentan$
LET wohnung$(1)=wohnung$(3) : REMark Harald in wohnung$(1)
LET wohnung$(3)=momentan$ : REMark Tom in wohnung$(3)
```

Mit dem folgenden Programm werden die zehn Golfer in einer Tabelle namens wohnung$ untergebracht. Die Namen der Bewohner werden mit ihren "Wohnungsnummern" (Tabellenindizes) ausgedruckt, um zu beweisen, daß sie dort wohnen. Nun tauschen die Bewohner der Wohnungen 1 und 3 ihre Wohnungen aus. Danach wird die Liste der Bewohner erneut ausgedruckt, um anzugeben, daß der Austausch stattgefunden hat.

```
100 REMark Wohnungen der Golfspieler
110 DIM wohnung$(10,7)
120 FOR zahl = 1 TO 10 : READ wohnung$(zahl)
130 liste_drucken
140 wohnung_wechseln
150 liste_drucken
160 REMark Ende des Hauptprogramms
170 DEFine PROCedure liste_drucken
180   FOR zahl = 1 TO 10 : PRINT zahl, wohnung$(zahl)
190 END DEFine
200 DEFine PROCedure wohnung_wechseln
210   LET momentan$ = wohnung$(1)
```

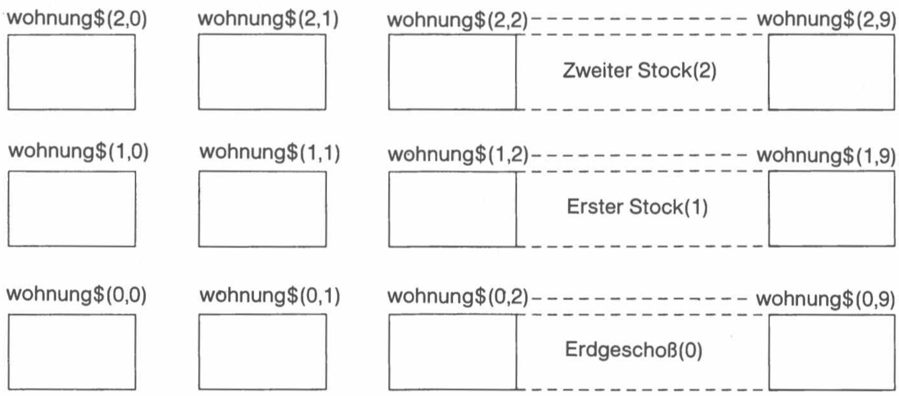
```
220   LET wohnung$(1) = wohnung$(3)
230   LET wohnung$(3) = momentan$
240 END DEFine
250 DATA ''Tom'', ''Karl'', ''Harald'', ''Jan'', ''Achim''
260 DATA ''Bert'', ''Jochen'', ''Walter'', ''Werner'',
        ''Dirk''
```

Ausgabe (Zeile 130)	Ausgabe (Zeile 150)
1 Tom	1 Harald
2 Karl	2 Karl
3 Harald	3 Tom
4 Jan	4 Jan
5 Achim	5 Achim
6 Bert	6 Bert
7 Jochen	7 Jochen
8 Walter	8 Walter
9 Werner	9 Werner
10 Dirk	10 Dirk

ZWEI-DIMENSIONALE TABELLEN

Gelegentlich ergeben sich aus der Natur eines Problems zwei Dimensionen, wie beispielsweise drei Stockwerke mit zehn Wohnungen anstatt nur einem Stockwerk mit 30 Wohnungen.

Angenommen, 20 oder mehr Golfer brauchen Wohnungen und es ist ein Block mit 30 Wohnungen auf drei Stockwerken mit jeweils zehn Wohnungen vorhanden. Dieser Block würde realistischerweise mit einer zweidimensionalen Tabelle dargestellt. Sie können sich die 30 Variablen wie folgt vorstellen:



Bei **DATA**-Anweisungen mit 30 Namen könnten die Namen folgendermaßen in die Wohnungen gesetzt werden:

```
120 FOR stock = 0 TO 2
130   FOR zahl = 0 TO 9
140     READ wohnung$(stock,zahl)
150   END FOR zahl
160 END FOR stock
```

Außerdem wird eine **DIM**-Anweisung benötigt:

```
20 DIM wohnung$(2,9,7)
```

Mit dieser Anweisung wird angegeben, daß der erste Index von 0 bis 2 (Stockwerknummer) und der zweite Index von 0 bis 9 (Wohnungsnummer) gehen kann. Mit der dritten Zahl wird die Höchstzahl von Zeichen in jedem Tabellenelement angegeben. Nun fügen wir einen Programmteil hinzu, der ausdruckt, daß die Golfer in den Wohnungen wohnen. Um Platz zu sparen, werden Buchstaben benutzt.

```
100 REMark 30 Golfspieler
110 DIM wohnung$(2,9,7)
120 FOR stock = 0 TO 2
130   FOR zahl = 0 TO 9
140     READ wohnung$(stock,zahl) : REMark Golfspieler geht rein
150   END FOR zahl
160 END FOR stock
170 REMark Ende der Eingabe
```

```
180 FOR stock = 0 TO 2
190   PRINT ''Stockwerk_nummer'' ! stock
200   FOR zahl = 0 TO 9
210     PRINT 'Wohnung' ! zahl ! wohnung$(stock,zahl)
220   END FOR zahl
230 END FOR stock
240 DATA ''A'',''B'',''C'',''D'',''E'',''F'',
       ''G'',''H'',''I'',''J''
250 DATA ''K'',''L'',''M'',''N'',
       ''O'',''P'',''Q'',''R'',''S'',''T''
260 DATA ''U'',''V'',''W'',''X'',''Y'',''Z'',
       ''Ä'',''Ö'',''Ü'',''ß''
```

Die Ausgabe beginnt mit:

```
Stockwerknummer 0
Wohnung 0 A
Wohnung 1 B
Wohnung 2 C
```

Insgesamt werden alle Bewohner angegeben.

TABELLEN-AUFTEILUNG

Dieser Abschnitt ist unter Umständen etwas schwierig zu lesen, obwohl es sich im Grunde genommen um dasselbe Konzept wie bei der String-Aufteilung handelt. Wahrscheinlich müssen Sie Teilstrings benutzen, wenn Sie über das Anfangsstadium der Programmierung hinausgehen möchten. Teiltabellen sind wesentlich seltener erforderlich, so daß Sie diesen Abschnitt besonders beim ersten Lesen vielleicht überspringen möchten.

Wir benutzen wieder das Beispiel mit den Wohnungen für die Golfer, um das Konzept der Aufteilung von Tabellen darzulegen. Die Wohnungen sind von 0 bis 9 numeriert, so daß wir mit einzelnen Ziffern arbeiten können. Aus Platzgründen werden die Namen mit einzelnen Zeichen angegeben.

	2,0	2,1	2,2	2,2	3,4	2,5	2,6	2,7	2,8	2,9
wohnung$	U	V	W	X	Y	Z	@	£	$	%

	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9
wohnung$	K	L	M	N	O	P	Q	R	S	T

	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
wohnung$	A	B	C	D	E	F	G	H	I	J

Die folgenden Angaben wählen jeweils einen der obigen Tabelle aus:

`wohnung$(1,3)` Stellt ein einzelnes Tabellenelement mit dem Wert N dar.

`wohnung$(1,1 TO 6)` Stellt sechs Elemente mit den Werten L M N O P Q dar.

Tabellenelement	Wert
wohnung$(1,1)	L
wohnung$(1,2)	M
wohnung$(1,3)	N
wohnung$(1,4)	O
wohnung$(1,5)	P
wohnung$(1,6)	Q

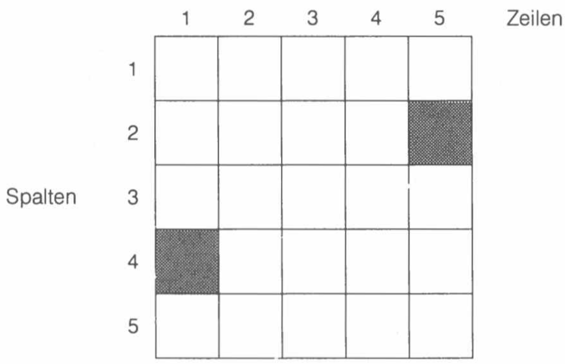
`wohnung$(1)` Bedeutet wohnung$(1,0 TO 9)
Zehn Elemente mit den Werten K L M N O P Q R S T

In diesen Beispielen wird ein Bereich von Werten für einen Index anstelle eines einzigen Wertes angegeben. Wird gar kein Index angegeben, so wird von dem ganzen Bereich ausgegangen. In dem dritten Beispiel fehlt der zweite Index. Er wird vom System als **0 TO 9** angenommen.

Die Techniken der Aufteilung von Tabellen und Strings sind gleich, auch wenn die letztere häufiger benutzt wird.

AUFGABEN ZU KAPITEL 13

1. **SORTIEREN**
   Besetzen Sie eine Tabelle mit zehn Zahlen vor, die Sie aus einer **DATA**-Anweisung auslesen. Durchsuchen Sie die Tabelle nach der niedrigsten Nummer. Machen Sie diese niedrigste Nummer zum Wert des ersten Elementes einer neuen Tabelle. Ersetzen Sie den Wert in der ersten Tabelle durch eine sehr große Zahl. Wiederholen Sie dieses Verfahren, indem die zweitniedrigste Zahl zum zweiten Wert in der neuen Tabelle gemacht wird usw., bis Sie eine sortierte Tabelle mit Zahlen haben, die dann ausgedruckt wird.
2. **LEITERSPIEL**
   Erstellen Sie ein Leiterspiel mit einer 100 Elemente umfassenden numerischen Tabelle. Jedes Element soll entweder:
   Null
   oder eine Zahl im Bereich von 10 bis 90 umfassen. Der Spieler muß zu dieser Zahl gehen, indem er auf der Leiter nach oben oder nach unten geht;
   oder die Ziffern 1, 2, 3, usw. um die jeweilige Position eines Spielers anzugeben.
   Erstellen Sie sechs Leitern, indem Sie die Zahlen in die Tabelle setzen und einen "Solo"-Lauf durch einen einzigen Spieler simulieren, mit dem das Spiel getestet wird.
3. **FELDER EINES KREUZWORTRÄTSELS**



   Kreuzworträtsel verfügen im allgemeinen über eine ungerade Zahl von Zeilen und Spalten. Die schwarzen Quadrate bilden ein symmetrisches Muster. Das Muster verfügt über eine Rotationssymmetrie, da es durch Drehung um 180 Grad nicht geändert wird.
   Nach der Drehung um 180 Grad könnte das Quadrat in Zeile 4, Spalte 1 zu dem Quadrat in Zeile 2, Spalte 5 werden. Das heißt, Zeile 4, Spalte 1 wird zu Zeile 2, Spalte 5 in einem 5 × 5 Raster.
   Schreiben Sie ein Programm, um ein derartiges symmetrisches Muster zu erzeugen und anzuzeigen.
4. Ändern Sie das Muster für das Kreuzworträtsel, so daß keine waagerechten oder senkrechten Folgen von weniger als vier weißen Quadraten mehr vorhanden sind.
5. **KARTEN MISCHEN**
   Die Karten haben die Nummern 1 bis 52 und sind in einer Tabelle gespeichert. Sie können bei Bedarf problemlos in tatsächliche Kartenwerte umgewandelt werden. Die Karten müssen wie folgt gemischt werden.
   - Wählen Sie eine beliebige Position im Bereich von 1 bis 51, z. B. 17.
   - "Merken" Sie sich Kartennummer und Tabellenplatz in einem Zwischenspeicher.
   - Schieben Sie alle Karten in den Positionen 52 bis 18 in die Positionen 51 bis 17.
   - Schreiben Sie die ausgewählte Kartennummer aus dem Zwischenspeicher in Position 52.
   - Gehen Sie mit den Bereichen 1 bis 50, 1 bis 49 usw. bis zu 1 bis 2 gleich vor, so daß das Kartenpaket gut gemischt ist.
   - Geben Sie das Ergebnis des Mischvorgangs aus.
6. Erstellen Sie sechs **DATA**-Anweisungen, wobei jede einen Vornamen, Initialen und eine Telefonnummer (Vorwahl und Durchwahl) enthält. Schaffen Sie eine entsprechende Tabellenstruktur zur Speicherung dieser Information und lesen Sie diese mit **READ** in die Tabellen ein.
   Drucken Sie die Daten mit einer separaten **FOR**-Schleife aus und erläutern Sie, warum das Eingabeformat (**DATA**), das interne Format (Tabellen) und das Ausgabeformat nicht unbedingt alle identisch sind.

KAPITEL 14
PROGRAMM-STRUKTUR

In diesem Kapitel wollen wir uns noch einmal mit der Programmsturktur befassen: Schleifen und Entscheidungen oder Auswahlen. Wir haben versucht, die Grundlagen so einfach wie möglich darzustellen. Allerdings ist SuperBASIC so aufgebaut, daß es sowohl mit der einfachsten als auch mit der komplexesten Ebene und allen dazwischenliegenden Ebenen fertig wird. Einige Teile dieses Abschnittes sind schwierig. Sollten Sie noch nicht sehr häufig programmiert haben, so werden Sie diese Teile unter Umständen überspringen. In diesem Abschnitt werden im einzelnen folgende Themen behandelt:

- Schleifen
- Verschachtelte Schleifen
- Einfache Entscheidungen
- Mehrfachentscheidungen.

Die letzteren Teile dieses ersten Abschnittes, Schleifen, werden schwierig. Dort wird dargelegt, wie SuperBASIC Probleme löst, die bei anderen Basic-Dialekten einfach ignoriert werden. Diese Teile können Sie gegebenenfalls überspringen. Die anderen Abschnitte sind jedoch einfacher.

SCHLEIFEN

In diesem Abschnitt werden wir bekannte Probleme der Wiederholung darstellen, indem wir einige Szenen aus dem Wilden Westen simulieren. Der Kontext mag weit hergeholt und trivial sein, bietet jedoch eine einfache Diskussionsgrundlage und stellt die Schwierigkeiten dar, die sich bei Programmierungsaufgaben ergeben.

BEISPIEL 1

In dem alten Schulhaus ist ein Bandit eingeschlossen. Der Sheriff hat sechs Patronen in seinem Gewehr. Simulieren Sie nun, wie diese sechs Schüsse abgegeben werden.

Programm 1

```
100 REMark FOR Western
110 FOR patronen = 1 TO 6
120   PRINT ''Zielen''
130   PRINT ''Schuß''
140 END FOR patronen
```

Programm 2

```
100 REMark REPeat Western
110 LET patronen = 6
120 REPeat bandit
130   PRINT ''Zielen''
140   PRINT ''Schuß''
150   LET patronen = patronen - 1
160   IF patronen = 0 THEN EXIT bandit
170 END REPeat bandit
```

Beide Programme führen zu derselben Ausgabe:

```
Zielen
Schuß
```

wird sechs Mal ausgedruckt.

Wird in jedem Programm die 6 in eine beliebige Zahl bis hinunter zu 1 geändert, werden beide Programme noch wie erwartet ausgeführt. Was geschieht jedoch, wenn das Gewehr leer ist, bevor Schüsse abgegeben werden?

BEISPIEL 2

Angenommen, jemand hat heimlich sämtliche Patronen aus dem Gewehr des Sheriffs herausgenommen. Was geschieht nun, wenn Sie die 6 einfach in jedem Programm in 0 ändern?

Programm 1

```
100 REMark Western : FOR 0 - Fall
110 FOR patronen = 1 TO 0
120   PRINT ''Zielen''
130   PRINT ''Schuß''
140 END FOR patronen
```

Dieses Programm arbeitet einwandfrei. Es kommt zu keiner Ausgabe. SuperBASIC verarbeitet den "Null-Fall" einwandfrei.

Programm 2

```
100 REMark Western : REPeat funktioniert nicht richtig
110 LET patronen = 0
120 REPeat bandit
130   PRINT ''Zielen''
140   PRINT ''Schuß''
150   LET patronen = patronen - 1
160   IF patronen = 0 THEN EXIT bandit
170 END REPeat bandit
```

Dieses Programm erfüllt sein Ziel in zweierlei Hinsicht nicht:

1. `Zielen`
   `Schuß`

   wird ausgedruckt, obwohl keine Patronen in dem Gewehr waren.

2. Zu dem Zeitpunkt, zu dem die Variable *Patronen*, in Zeile 160 getestet wird, hat sie den Wert −1 und wird danach niemals zu Null. Das Programm führt eine endlose Schleife aus. Dieses Problem der endlosen Schleifen kann behoben werden, indem Zeile 160 neu geschrieben wird:

   ```
   160 IF patronen < 1 then EXIT bandit
   ```

Hier liegt ein Programmierungsfehler vor, da der "Null-Fall" nicht berücksichtigt wird. Dieser Fehler kann behoben werden, indem ein bedingter **EXIT**-Befehl vor die **PRINT**-Anweisung gestellt wird.

Programm 3

```
100 REMark Western : REPeat 0-Fall
110 LET patronen = 0
120 REPeat bandit
130   IF patronen = 0 THEN EXIT bandit
140   PRINT ''Zielen''
150   PRINT ''Schuß''
160   LET patronen = patronen - 1
170 END REPeat bandit
```

Dieses Programm arbeitet nun einwandfrei, unabhängig von dem Ausgangswert der Patronen, solange es sich hier um eine positive ganze Zahl oder um Null handelt. Methode 2 entspricht der **REPEAT . . . UNTIL**-Schleife einiger Sprachen. Methode 3 entspricht der **WHILE . . . ENDWHILE**-Schleife einiger Sprachen. **REPeat . . . END REPeat** mit **EXIT** ist jedoch flexibler als eine dieser beiden Methoden oder eine Kombination der beiden Methoden.

Haben Sie andere BASIC-Versionen benutzt, so werden Sie sich unter Umständen fragen, was mit der **NEXT**-Anweisung geschehen ist. Wir werden in Kürze auf sie eingehen. Sie werden jedoch feststellen, daß beide Schleifen über eine ähnliche Struktur verfügen und beide benannt sind.

FOR *Name* =	(Öffnender Befehl)	**REPeat** *Name*
(Anweisungen)	(Inhalt)	(Anweisungen)
END FOR Name	(Schließender Befehl)	**END REPeat** *Namen*

Die **REPeat**-Schleife muß normalerweise zusätzlich über eine **EXIT**-Anweisung verfügen, ansonsten wird sie niemals beendet.

Beachten Sie bitte, daß nach Ausführung einer **EXIT**-Anweisung das Programm nach der Anweisung fortgesetzt wird, die unmittelbar hinter der **END**-Anweisung der Schleife steht.

Eine **NEXT**-Anweisung kann in eine Schleife gestellt werden. Nach ihrer Ausführung wird das Programm bei der Anweisung fortgesetzt, die direkt hinter dem öffnenden Befehl **FOR** oder **REPeat** steht. Im Falle einer **FOR**-Schleife wird die Steuervariable vorher fortgeschaltet. Sie kann als Gegenstück zu der **EXIT**-Anweisung betrachtet werden.

BEISPIEL 3

Die Situation ist dieselbe wie in Beispiel 1. Der Sheriff verfügt über ein Gewehr mit sechs Patronen und er muß auf den Banditen schießen. Allerdings gibt es hier noch zwei weitere Bedingungen:

1. Trifft er den Banditen, so schießt er nicht weiter und kehrt nach Dodge City zurück.
2. Hat er keine Patronen mehr, bevor er den Banditen getroffen hat, so bittet er seinen Partner, auf den Banditen aufzupassen, während er (der Sheriff) nach Dodge City zurückkehrt.

Programm 1

```
100 REMark Western : FOR mit Schleifennachsatz
110 FOR patronen = 1 TO 6
120   PRINT ''Zielen''
130   PRINT ''Schuß''
140   LET treffer = RND(9)
150   IF treffer = 7 THEN EXIT patronen
160 NEXT patronen
170   PRINT ''Paß auf den Banditen auf''
180 END FOR patronen
190 PRINT ''Zurück nach Dodge City''
```

In diesem Fall stellt der Inhalt zwischen **NEXT** und **END FOR** eine Art Nachwort dar, das nur ausgeführt wird, wenn die **FOR**-Schleife voll zu Ende geführt wird. Kommt es zu einem vorzeitigen **EXIT**, so wird das Nachwort nicht ausgeführt.

Derselbe Effekt kann mit einer **REPeat**-Schleife erzielt werden, auch wenn dies nicht unbedingt der beste Weg ist. Es lohnt sich jedoch, diese Schleife (vielleicht beim zweiten Lesen) genauer zu betrachten, wenn Sie Strukturen verstehen möchten, die einfach genug für einfache Lösungen und leistungsfähig genug für eventuell auftretende komplexe Situationen sind.

Programm 2

```
100 REMark Western : REPeat mit Schleifennachsatz
110 LET patronen = 6
120 REPeat bandit
130   PRINT ''Zielen''
140   PRINT ''Schuß''
150   LET treffer = RND(9)
160   IF treffer = 7 THEN EXIT bandit
170   LET patronen = patronen-1
180   IF patronen = 0 THEN EXIT bandit
190   PRINT ''Paß auf den Banditen auf''
200 END REPeat bandit
210 PRINT ''Zurück nach Dodge City''
```

Dieses Programm arbeitet einwandfrei, solange der Sheriff am Anfang mindestens eine Patrone hat. Es wird nicht richtig ausgeführt, wenn Zeile 110 folgendermaßen aussieht:

```
110 LET patronen = 0
```

Sicher wäre der Sheriff dumm, ein Unternehmen dieser Art zu starten, wenn er gar keine Patronen hat. Wir wollen hier besprechen, wie eine gute Struktur in der komplexesten Situation bewahrt werden kann. Und hierzu haben wir eine einfache Ausgangssituation ausgewählt. Wir wissen, was getan werden soll. Komplexe strukturelle Probleme entstehen im allgemeinen in Kontexten, die schwieriger sind, als unsere Simulation des Wilden Westens. Wird jedoch eine Lösung für das Problem gesucht, bei dem ein möglicher Treffer, das Verschießen der Patronen und ein Nachwort, sowie der Null-Fall berücksichtigt werden, so muß die folgende Zeile zu dem obigen Programm hinzugefügt werden:

```
125 IF patronen = 0 then PRINT ''Paß auf den Banditen auf'':
   EXIT bandit
```

Wir können uns kein komplexeres Problem als dieses mit einer einzigen Schleife vorstellen. SuperBASIC kann diese Aufgabe problemlos lösen.

VERSCHACHTELTE SCHLEIFEN

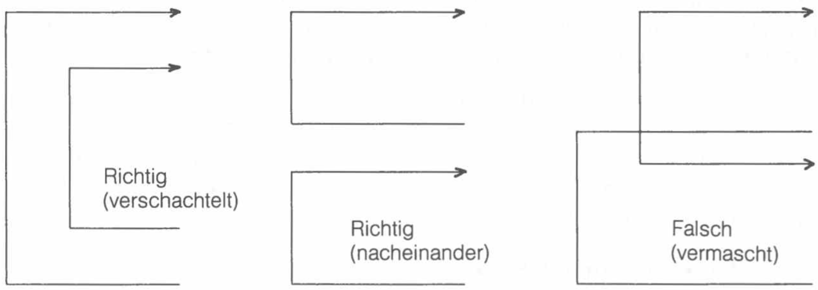
Betrachten Sie die folgende **FOR**-Schleife, mit der eine Reihe von Punkten in verschiedenen, zufällig ausgewählten Farben (mit Ausnahme von Schwarz) gezeichnet wird.

```
100 REMark Punktmusterreihe
110 PAPER 0 : CLS
120 LET senk = 50
130 FOR waag = 20 TO 60
140   INK RND(2 TO 7)
150   POINT waag, senk
160 END FOR waag
```

Dieses Programm zeichnet eine Reihe von Punkten:

. . . .

Möchten Sie beispielsweise 51 Punktreihen, so müssen Sie eine Reihe von Werten ab 30 bis 80 zeichnen. Hier muß jedoch stets die Regel beachtet werden, daß eine Struktur vollständig in eine andere Struktur gesetzt werden oder um die andere Struktur herumgehen kann. Sie kann auch folgerichtig folgen, kann jedoch nicht mit einer anderen Struktur "vermascht" werden. In Büchern über die Programmierung wird die Beziehung zwischen **FOR**-Schleifen häufig mit folgenden Diagrammen dargestellt:



In SuperBASIC gilt diese Regel für sämtliche Strukturen. Sie können sämtliche Probleme lösen, indem Sie sie richtig anwenden. Demzufolge behandeln wir die **FOR**-Schleife als Gesamtheit und entwerfen ein neues Programm:

```
FOR senk = 30 TO 80
  ┌──────────────────────────┐
  │ FOR waag = 20 TO 60      │
  │   INK RND (2 To 7)       │
  │     POINT waag, senk     │
  │   END FOR waag           │
  └──────────────────────────┘
END FOR senk
```

Übersetzen wir diesen Entwurf in ein Programm, so können wir nicht nur erwarten, daß es einwandfrei ausgeführt wird, sondern wissen auch, was es tut. Es zeichnet ein Rechteck, das aus Punktmusterreihen besteht.

```
100 REMark Punktmusterreihen
110 PAPER 0 : CLS
120 FOR senk = 30 TO 80
130   FOR waag = 20 TO 60
140     INK RND(2 TO 7)
150     POINT waag, senk
160   END FOR waag
170 END FOR senk
```

Verschiedene Strukturen können verschachtelt werden. Angenommen, wir ersetzen die innere **FOR**-Schleife des obigen Programms durch eine **REPeat**-Schleife. Wir beenden die **REPeat**-Schleife, wenn der Farbcode Null bei einer Auswahl in dem Bereich Null bis Sieben auftritt.

```
100 REMark REPeat in einer FOR Schleife
110 PAPER 0 : CLS
120 FOR senk = 30 TO 80
130   LET waag = 19
140   REPeat punkte
150     LET farbe = RND(7)
160     INK farbe
170     LET waag = waag + 1
180     POINT waag, senk
190     IF farbe = 0 THEN EXIT punkte
200   END REPeat punkte
210 END FOR senk
```

Ein Großteil des Wissens über die Programmsteuerung und die Programmstruktur kann in zwei Regeln ausgedrückt werden:

1. Erstellen Sie Ihr Programm nur mit den zulässigen Strukturen für Schleifen und die Entscheidungsfällung.
2. Jede Struktur muß entweder vollständig auf eine andere Struktur folgen oder ganz in einer anderen Struktur enthalten sein.

EINFACHE ENTSCHEIDUNGEN

Es gibt drei Arten einfacher Entscheidungen, die wir anhand des Beispiels, was wir bei Regen tun, einfach darlegen können.

a)
```
100 REMark Kurze Version von IF
110 LET regen = RND(0 TO 1)
120 IF regen THEN PRINT ''Öffnen Sie Ihren Schirm''
```
b)
```
100 REMark Lange Version von IF...END IF
110 LET regen = RND(0 TO 1)
120 IF regen THEN
130   PRINT ''Ziehen Sie Ihren Mantel an''
140   PRINT ''Öffnen Sie Ihren Schirm''
150   PRINT ''Gehen Sie schneller''
160 END IF
```
c)
```
100 REMark Lange Version von IF...ELSE...END IF
110 LET regen = RND(0 TO 1)
120 IF regen THEN
130   PRINT ''Fahren Sie mit dem Bus''
140 ELSE
150   PRINT ''Gehen Sie zu Fuß''
160 END IF
```

All diese Beispiele sind Binärentscheidungen; d. h. es wird eine von zwei verschiedenen Möglichkeiten ausgewählt. Die beiden ersten Beispiele sind einfach: entweder es geschieht etwas oder es geschieht nichts. Das dritte Beispiel ist die allgemeinste Form einer Binärentscheidung mit zwei verschiedenen Möglichkeiten.

In den langen Formen kann **THEN** weggelassen werden.

In der kurzen Form kann **THEN** durch **:** ersetzt werden.

BEISPIEL

Hier wollen wir ein komplexeres Beispiel betrachten, in dem es natürlich scheint, Binärentscheidungen zu verschachteln. Diese Art der Verschachtelung kann verwirrend sein und Sie sollten dies nur tun, wenn die Aufgabenstellung es nahelegt. Hier muß der Darstellungsform, insbesondere der Einrückung, besondere Aufmerksamkeit gewidmet werden.

Analysieren Sie ein Textstück, um die Anzahl von Vokalen, Konsonanten und anderen Zeichen zu zählen. Leerzeichen werden ignoriert. Der Einfachheit halber wird der Text in Großbuchstaben angegeben.

Daten

"COMPUTER-GESCHICHTE WURDE 1984 GEMACHT"

Entwurf

Die Daten werden eingelesen:

```
    FOR jedes Zeichen
      IF Buchstabe THEN
        IF Vokal THEN
          Vokale – 1
        ELSE
          Konsonanten + 1
        END IF
      ELSE
    IF keine Leertaste THEN andere + 1
END IF
END FOR
PRINT Ergebnisse
```

Programm

```
100 REMark Zählt die verschiedenen Zeichen
110 RESTORE 290
120 READ text$
130 LET vokale = 0 : kons = 0 : andere = 0
140 FOR zahl = 1 TO LEN(text$)
150   LET zeich$ = text$(zahl)
160   IF zeich$ >= ''A'' AND zeich$ <= ''Z''
170     IF zeich$ INSTR ''AEIOU''
180       LET vokale = vokale + 1
190     ELSE
200       LET kons = kons + 1
210     END IF
220   ELSE
```

```
230    IF zeich$ <> '' '' THEN andere = andere + 1
240   END IF
250 END FOR zahl
260 PRINT ''Die Anzahl der Vokale beträgt'' ! vokale
270 PRINT ''Die Anzahl der Konsonanten beträgt'' ! kons
280 PRINT ''Die Anzahl der anderen Zeichen beträgt'' ! andere
290 DATA ''COMPUTER-GESCHICHTE WURDE 1984 GEMACHT''
```

Ausgabe

```
Die Anzahl der Vokale beträgt 10
Die Anzahl der Konsonanten beträgt 20
Die Anzahl der anderen Zeichen beträgt 5
```

MEHRFACH-ENTSCHEIDUNGEN – SELect

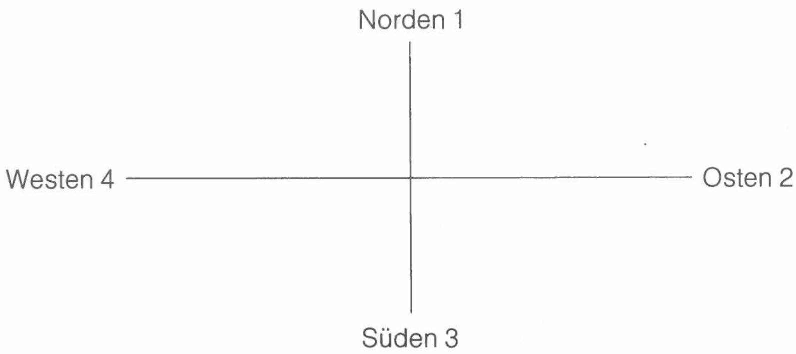
In den Fällen, in denen es drei oder mehr mögliche Aktionen gibt und keine von einer vorherigen Auswahl abhängt, wird **SELect** benutzt. Mit diesem Befehl kann eine Auswahl aus einer beliebigen Anzahl von Möglichkeiten getroffen werden.

BEISPIEL

Eine Zauberschlange wächst endlos, indem an ihrem Kopf immer wieder ein Abschnitt hinzugefügt wird. Jeder Abschnitt kann eine Länge von bis zu 20 Einheiten und eine neue Farbe oder die alte Farbe aufweisen. Jeder neue Abschnitt muß in eine der Richtungen Norden, Süden, Osten oder Westen wachsen. Die Schlange beginnt in der Mitte des Fensters.

Methode

Während die Schlange noch auf dem Bildschirm angezeigt wird, wählen Sie eine zufällige Länge und Schriftfarbe. Die Richtung kann mit einer Zahl 1, 2, 3 oder 4 wie dargestellt ausgewählt werden.



Entwurf

Hintergrundfarbe wählen.

Setzen Sie die Schlange in die Mitte des Fensters.
REPeat
  Wählen Sie Richtung, Farbe und Wachstumslänge.
  FOR Abschnitt = 1 **TO** Wachsen
    Die Schlange in Richtung Norden, Süden, Osten
    oder Westen wachsen lassen.
  IF Schlange nicht im Fenster **THEN EXIT**
END FOR
END REPeat
PRINT Abschließende Meldung

Programm

```
100 REMark Zauberschlange
110 PAPER 0 : CLS
120 LET waag = 50 : senk = 50
130 REPeat schlange
140   LET richtung = RND(1 TO 4) : farbe = RND(2 TO 7)
150   LET wachsen = RND(2 TO 20)
160   INK farbe
170   FOR abschnitt = 1 TO wachsen
180     SELect ON richtung
190       ON richtung = 1
200         LET senk = senk + 1
210       ON richtung = 2
220         LET waag = waag + 1
230       ON richtung = 3
```

```
240        LET senk = senk -1
250       ON richtung = 4
260       LET waag = waag-1
270      END SELect
280      IF waag<1 OR waag>99 OR senk<1 OR senk>99 THEN EXIT
          schlange
290      POINT waag,senk
300    END FOR abschnitt
310 END REPeat schlange
320 PRINT ''Schlange außerhalb der Kante''
```

Die Syntax der **SELect ON**-Struktur ermöglicht auch die Auswahl aus einer Liste von Werten, wie beispielsweise:

```
5,6,8,10 TO 13
```

Außerdem kann auch die Ausführung einer Aktion vorgesehen werden, wenn keiner der angegebenen Werte gefunden wird. Die vollständige Struktur weist die nachstehende Form auf.

LANGE FORM

```
SELect ON num
ON num = Werteliste
  Anweisungen
ON num = Werteliste
  Anweisungen
  –
  –
  –
  –
ON num = REMAINDER
  Anweisungen
END SELect
```

num muß der Name einer beliebigen numerischen Variablen sein. Die Anweisung **ON** num = **REMAINDER** kann entfallen, wenn sie nicht benötigt wird.

KURZE FORM

Es gibt auch eine kurze Form für die **SELect**-Struktur. Zum Beispiel:

```
100 INPUT zahl
110 SELect ON zahl = 0 TO 9 : PRINT ''Ziffer''
```

wird wie erwartet ausgeführt.

AUFGABEN ZU KAPITEL 14

1. 10 Zahlen werden in einer Tabelle gespeichert und sortiert. Die geschieht, indem das erste Zahlenpaar verglichen und gegebenenfalls beide Zahlen vertauscht werden. Anschließend wird mit dem zweiten Paar (der zweiten und dritten Zahl), usw. ebenso verfahren. Nach dem neunten Schritt ist gewährleistet, daß die höchste Zahl in der richtigen Position steht. Weitere acht Folgen garantieren acht weitere richtige Positionen, wobei nur noch die niedrigste Zahl übrig bleibt, die in der einzigen (richtigen) Position stehen muß. Die einfachste Form dieses Sortiervorgangs von zehn Zahlen fordert neun Folgen von neun Vergleichen.

2. Suchen Sie nach Möglichkeiten, diesen Sortiervorgang zu beschleunigen. Erwarten Sie jedoch nicht, daß diese sehr effizient sind.

3. Ein Auktionator möchte eine alte Uhr versteigern, wobei als Mindesteinsatz 50 PDM gefordert werden. Bietet niemand, so kann er auf 40, 30, 20 DM, jedoch nicht weiter, heruntergehen, um die Versteigerung in Gang zu setzen. Bietet niemand, so wird die Uhr zurückgezogen. Nachdem die Versteigerung begonnen hat, sind nur Erhöhungen von jeweils 5 DM zulässig, bis das endgültige Angebot gemacht wird. Ist das letzte Angebot 35 DM (der "Reservepreis") oder mehr, so wird die Uhr verkauft. Ansonsten wird sie vom Verkauf zurückgezogen.

   Simulieren Sie die Versteigerung, indem Sie einen sechsseitigen Würfel für den Beginn der Gebote benutzen. Eine "Sechs" bei einem der Anfangspreise startet die Auktion. Natürlich simulieren Sie den Würfel im Computer.

   Nach Beginn der Gebote muß eine Drei zu Vier-Chance eines höheren Gebotes bei jeder Aufforderung bestehen.

4. Bei einer Schießerei im Wilden Westen hat der Sheriff keine Munition mehr und möchte einen bewaffneten Verbrecher verhaften, der sich in einem Wald versteckt hält. Er reitet in Richtung des Waldes und hofft, daß der Verbrecher schießt. Nachdem sechs Schüsse abgegeben wurden, hofft er, daß er in den Wald gehen und den Verbrecher überwältigen kann, während dieser versucht, sein Gewehr neu zu laden. Simulieren Sie diese Begegnung, indem Sie dem Verbrecher eine Chance von 1 zu 20 geben, den Sheriff mit jedem Schuß zu treffen. Wurde der Sheriff nach sechs Schüssen nicht getroffen, so verhaftet er den Verbrecher.

5. Der Sheriff gibt seinem Stellvertreter folgende Instruktionen:

   "Ist das Gewehr leer, so laden Sie es erneut. Ist es nicht leer, so schießen Sie weiter, bis Sie den Verbrecher getroffen haben oder dieser aufgibt. Wenn Mexico Pete auftaucht, so laufen Sie schnell weg."

   Schreiben Sie ein Programm, in dem die folgenden Situationen berücksichtigt werden.

   - Was immer geschieht, gehen Sie nach Dodge City zurück.
   - Taucht Mexico Pete auf, so gehen Sie sofort zurück.
   - Ist das Gewehr leer, so laden Sie es erneut.
   - Ist das Gewehr nicht leer, so fordern Sie den Verbrecher auf, sich zu ergeben.
   - Ergibt sich der Verbrecher, so verhaften Sie ihn.
   - Ergibt er sich nicht, so geben Sie einen Schuß ab.
   - Ist der Verbrecher getroffen, so verhaften Sie ihn und versorgen seine Verletzung.

   In diesem Programm wird von einer unbegrenzten Anzahl von Patronen ausgegangen. Benutzen Sie einen simulierten "20seitigen Würfel". "7" bedeutet "Ergeben" und "13" bedeutet, daß der Verbrecher getroffen ist.

KAPITEL 15
PROZEDUREN UND FUNKTIONEN

Im ersten Teil dieses Abschnittes erläutern wir die einfacheren Aspekte der SuperBASIC Prozeduren und Funktionen. Dies geschieht anhand sehr einfacher Beispiele, damit Sie die einzelnen Aspekte problemlos verstehen können. Obwohl die Beispiele etwas konstruiert sind, werden Sie sehen, daß die Ideen auch in komplexeren Situationen benutzt werden können, wenn Sie sie erst einmal verstanden haben.

Auf diesen ersten Teil folgt dann ein Abschnitt mit der Bezeichnung "Warum Prozeduren?". Haben Sie die Ausführungen bis zu diesem Punkt mehr oder weniger verstanden, so haben Sie keine Probleme und können Prozeduren und Funktionen immer problemloser benutzen.

Mit SuperBASIC können Sie die einfacheren Dinge erst einmal auf einfache Art und Weise ausführen. Danach hat SuperBASIC noch andere Mittel, falls diese gewünscht werden. Zusätzliche Möglichkeiten und einige technische Fragen werden im zweiten Teil dieses Abschnittes beschrieben. Diese können jedoch, zumindest beim ersten Lesen, übergangen werden. Dennoch befinden Sie sich in einer stärkeren Position als die meisten Benutzer älterer BASIC-Versionen.

EINGABE-PARAMETER

In den vorhergehenden Abschnitten haben Sie gesehen, wie ein Wert an eine Prozedur übergeben werden kann. Hier ein weiteres Beispiel.

BEISPIEL

In "Chan's Chinarestaurant" stehen nur sechs Gerichte auf der Speisekarte.

Reisgerichte	Süßspeisen
1 Langusten	4 Eis
2 Hühnchen	5 Pfannkuchen
3 Spezial	6 Früchte

Chan berechnet seine Preise auf einfache Art und Weise. Er arbeitet mit Pfennigen und hat folgende Preise:

- Für ein Reisgericht 900 + 10 mal Menü-Nummer.
- Für Süßspeisen 120 mal Menü-Nummer.

Ein Gast, der den Spezial-Reis und ein Eis gegessen hat, würde also folgendes bezahlen:

```
900 + 10 * 3 + 120* 4 = 1410 Pfennig
```

Eine Prozedur, *Posten*, benötigt eine Menü-Nummer als Eingabe-Parameter und druckt die Kosten aus.

Programm

```
100 REMark Kosten für ein Gericht
110 posten 3
120 posten 4
130 DEFine PROCedure posten(nummer)
140   IF nummer <= 3 THEN LET preis = 900 + 10 * nummer
150   IF nummer >= 4 THE LET preis = 120 * nummer
160   PRINT ! preis !
170 END DEFine
```

Ausgabe

```
930 480
```

In dem Hauptprogramm wird die Prozedur mit den aktuellen Parametern 3 und 4 aufgerufen. Der formale Parameter *nummer* in der Prozedurdefinition, nimmt bei der Ausführung der Prozedur den Wert an, der aus dem Hauptprogramm übergeben wurde; in unserem Fall zuerst 3 und beim zweiten Aufruf 4. Beachten Sie, daß die formalen Parameter in Klammern stehen müssen, während dies bei den aktuellen Parametern nicht erforderlich ist.

BEISPIEL

Stellen Sie sich nun vor, daß auch die Variable in dem Hauptprogramm benutzt wurde. Mit ihr wird jedoch ein anderer Preis angegeben, beispielsweise der Preis für ein Glas Bier, 320 Pfennig. Das folgende Programm führt nicht zu dem gewünschten Ergebnis.

Programm

```
100 REMark Globale Variable 'preis'
110 LET preis = 320
120 posten 3
130 posten 4
140 PRINT ! preis !
150 DEFine PROCedure posten(nummer)
160   IF nummer <= 3 THEN LET preis = 900 + 10 * nummer
170   IF nummer >= 4 THEN LET preis = 120 * nummer
180 PRINT ! preis !
190 END DEFine
```

Ausgabe

```
930 480 480
```

Der Preis für das Bier wurde von der Prozedur geändert. Die Variable *preis*, wird als *globale Variable* bezeichnet, da sie an beliebiger Stelle im Programm benutzt werden kann.

BEISPIEL

Wenn die Variable *preis* zur lokalen Variablen erklärt wird, dann erkennt das Programm zwei verschiedene Variablen mit dem Namen *preis*. Es kennt sie allerdings nicht gleichzeitig. Solange die Prozedur ausgeführt wird, versteht es unter *preis* die lokale Variable. Die vorher im Hauptprogramm benutzte Variable *preis* wird dabei nicht verändert. Wenn die Prozedur beendet ist, wird die lokale Varialbe *preis* mitsamt dem Wert, der ihr zugewiesen wurde, vollständig vergessen. Wenn Sie jetzt wieder *preis* benutzen, dann versteht das Programm darunter die Variable, in der Sie den Preis für ein Glas Bier gespeichert haben.

Programm

```
100 REMark Lokale Variable 'preis'
110 LET preis = 320
120 posten 3
130 posten 4
140 PRINT ! preis !
150 DEFine PROCedure posten(nummer)
160   LOCal preis
170   IF nummer <= 3 THEN LET preis = 900 + 10 * nummer
180   IF nummer >= 4 THEN LET preis = 120 * nummer
190   PRINT ! preis !
200 END DEFine
```

Ausgabe

```
930 480 320
```

Diesmal wurde das Programm richtig ausgeführt. Durch Zeile 160 wird die Variable, *preis*, intern als nur zu der Prozedur, *Posten*, gehörig markiert. Die andere Variable, *preis*, ist davon nicht betroffen. Wie Sie sehen, sind lokale Variablen sehr nützlich.

BEISPIEL

Lokale Variable sind so nützlich, daß wir die formalen Parameter von Prozeduren automatisch zu lokalen Variablen machen. Auch wenn wir es vorher noch nicht erwähnt haben, können Parameter, wie beispielsweise *nummer* in den obigen Programmen nicht mit den Variablen des Hauptprogramms kollidieren. Um dies zu beweisen, lassen wir die **LOCAL**-Anweisung aus dem obigen Programm weg und benutzen *nummer* für den Bierpreis. Da *nummer* in der Prozedur lokal ist, gibt es keine Probleme.

Programm

```
100 REMark Lokaler Parameter
110 LET nummer = 320
120 posten 3
130 posten 4
140 PRINT ! nummer !
150 DEFine PROCedure posten(nummer)
160   LOCal preis
170   IF nummer <= 3 THEN LET preis = 900 + 10 * nummer
180   IF nummer >= 4 THEN LET preis = 120 * nummer
190   PRINT ! preis !
200 END DEFine
```

Ausgabe

```
930 480 320
```

AUSGABE-PARAMETER

Bis jetzt haben wir Parameter nur dazu benutzt, um Werte an die Prozedur zu übergeben. Angenommen, die Kosten eines Postens müssen wieder an das Hauptprogramm übergeben werden, damit eine Gesamtrechnung aufgestellt werden kann. Dies kann problemlos über einen anderen Parameter in dem Prozeduraufruf erfolgen. Als aktueller Parameter muß eine Variable eingesetzt werden, da sie einen Wert aus der Prozedur empfangen muß. Dies bezeichnen wir als Ausgabe-Parameter. Für diesen Parameter muß ein entsprechender variabler Parameter in der Prozedurdefinition vorhanden sein.

BEISPIEL

Benutzen Sie aktuelle Ausgabe-Parameter, *kosten_1* und *kosten_2*, um die Werte der Variablen, *preis*, aus der Prozedur aufzunehmen. Das Hauptprogramm berechnet die Gesamtrechnung und druckt sie aus.

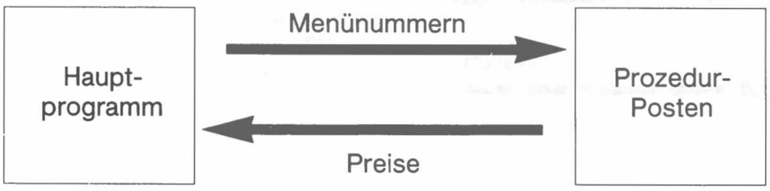
Programm

```
100 REMark Ausgabeparameter
110 LET nummer = 320
120 posten 3, kosten_1
130 posten 4, kosten_2
140 LET rechnung = nummer + kosten_1 + kosten_2
150 PRINT rechnung
160 DEFine PROCedure posten(nummer,kosten)
170   IF nummer <= 3 THEN LET kosten = 900 + 10 * nummer
180   IF nummer >= 4 THEN LET kosten = 120 * nummer
190 END DEFine
```

Ausgabe

```
1730
```

Die Parameter, *nummer* und *preis*, sind beide automatisch lokal, so daß es zu keinen Problemen kommen kann. In den Diagrammen wird dargestellt, wie Informationen aus dem Hauptprogramm in die Prozedur und wieder zurückübergeben werden.



Damit fürs Erste genug von Prozeduren und Parametern.

FUNKTIONEN

Sie wissen schon, wie eine Systemfunktion benutzt wird. So berechnet beispielsweise die Funktion:

```
SQRT(9)
```

den Wert 3, d.h. die Quadratwurzel von 9. Die Funktion gibt den Wert 3 zurück. Eine Funktion kann wie eine Merkmal einer Funktion besteht jedoch darin, daß sie genau einen Wert zurückgibt. Dies bedeutet, daß Sie die Funktion in schon vorhandenen Ausdrücken benutzen können. So können Sie:

```
PRINT 2*SQRT(9)
```

eingeben und als Ausgabe 6 erhalten. Somit verhält sich eine Funktion wie eine Prozedur mit einem oder mehreren Eingabe-Parametern. Der Funktionsname selbst der den zurückgegebenen Wert enthält, kann als Ausgabe-Parameter aufgefaßt werden.

Die Parameter brauchen keine numerischen Parameter zu sein.

```
LEN("string")
```

verfügt über einen String-Parameter, gibt jedoch den numerischen Wert 6 zurück.

BEISPIEL

Schreiben Sie das Programm des letzten Abschnitts neu, in dem preis als Ausgabe-Parameter benutzt wurde. Preis soll nun der Name der Funktion sein.

Der zurückzugebende Wert wird durch die **RETurn**-Anweisung dem Funktionsnamen zugewiesen, wie folgendes Beispiel zeigt.

Programm

```
100 REMark FuNktion mit RETurn
110 LET nummer = 320
120 LET rechnung = nummer + preis(3) + preis(4)
130 PRINT rechnung
140 DEFine FuNction preis(nummer)
150   IF nummer <= 3 THEN RETurn 900 + 10 * nummer
160   IF nummer >= 4 THEN RETurn 120 * nummer
170 END DEFine
```

Ausgabe

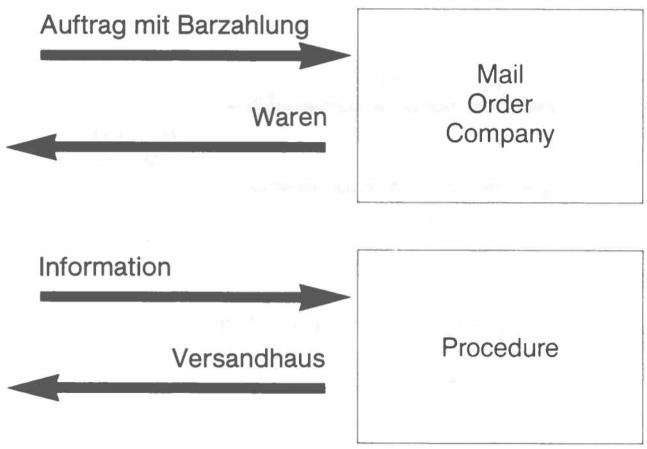
```
1730
```

Beachten Sie, wie viel einfacher der Aufruf von Funktionen im Vergleich zu Prozeduraufrufen ist.

WARUM PROZEDUREN?

Letzten Endes kann eine Prozedur als "Black Box" verstanden werden, die bestimmte Informationen von "außen"empfängt und bestimmte Operationen ausführt, in deren Rahmen auch bestimmte Informationen wieder nach "außen" zurückgesendet werden können. Mit "außen" kann das Hauptprogramm oder eine andere Prozedur gemeint sein.

Der Ausdruck "Black Box" impliziert, daß die interne Arbeitsweise keine Rolle spielt. Sie denken nur daran, was in diese "Black Box" hineingeht und was aus ihr herauskommt. Benutzt eine Prozedur beispielsweise eine Variable, *zahl*, und ändert sie deren Wert, so kann sich dies auf eine Variable desselben Namens im Hauptprogramm auswirken. Denken Sie an ein Versandhaus. Sie senden einen Auftrag und das Bargeld an das Versandhaus, das Ihnen dann die Ware schickt. Informationen werden an eine Prozedur gesendet, die dann eine bestimmte Aktion ausführt oder neue Informationen zurücksendet.



Dies wäre eine unerwünschte Nebenwirkung. Außerdem möchten Sie nicht, daß eine Prozedur nicht geplante Änderungen an Werten von Variablen vornimmt, die in dem Hauptprogramm benutzt werden.

Selbstverständlich können Sie verhindern, daß Variablennamen in einem Programm nicht zweimal benutzt werden. Dies geht bis zu einem gewissen Punkt. Wir haben jedoch in diesem Kapitel gezeigt, wie Probleme vermieden werden, selbst wenn Sie vergessen, welche Variablen in einer bestimmten Prozedur benutzt wurden.

Als zweites werden Prozeduren dazu benutzt, ein Programm modular aufzubauen. Anstatt ein langes Hauptprogramm zu benutzen, können Sie den Job in Teile aufteilen, die Seymour Papert, der Erfinder von **LOGO**, als "geistesgerechte Bissen" bezeichnet hat. Eben dies sind die Prozeduren, wobei jede klein genug ist, um problemlos verstanden und benutzt werden zu können. Sie sind durch die Prozeduraufrufe in einer Folge oder Hierarchie miteinander verbunden.

Als drittes soll mit Prozeduren verhindert werden, daß eine Folge von Anweisungen mehrmals geschrieben werden muß. Schreiben Sie sie einmal als Prozedur und rufen Sie sie gegebenenfalls zweimal auf. Funktionen und Prozeduren, die für ein Programm geschrieben wurden, können häufig direkt ohne Änderung von anderen Programmen benutzt werden. Auf diese Weise kann eine Bibliothek mit häufig benutzten Prozeduren und Funktionen erstellt werden.

Nachfolgend ein weiteres Beispiel, in dem dargelegt wird, wie Prozeduren ein Programm modular gestalten.

BEISPIEL

In Chans China-Restaurant werden sechs Gerichte bestellt, wobei die Speisekarte folgende Gerichte umfaßt:

Posten-Nummer	Gericht	Preis
1	Garneelen	3.50
2	Hühnchen	2.80
3	Spezial	3.30

Nun schreiben Sie Prozeduren für die folgenden Aufgaben:

1. Erstellen Sie zwei Tabellen mit jeweils drei Elementen, in denen das Menü, die Gerichte und Preise gespeichert werden. Benutzen Sie eine **DATA**-Anweisung.
2. Simulieren Sie eine Bestellung für sechs zufällig ausgewählte Gerichte mit Hilfe einer Prozedur. Prüfen Sie, wie oft jedes Gericht ausgewählt wurde.
3. Übergeben Sie die drei Zahlen an eine Prozedur, *kellner*, die wiederum die Kosten der Bestellung mit Hilfe eines Parameters *kosten* an das Hauptprogramm zurückgibt. Die Prozedur, *kellner*, ruft zwei weitere Prozeduren, *rechnen* und *koch* auf. Sie berechnen die Kosten und simulieren das "Kochen".
4. Die Prozedur *koch*, druckt einfach die erforderliche Zahl und den Namen jedes Gerichtes aus.

Das Hauptprogramm ruft Prozeduren nach Bedarf auf, ermittelt die Gesamtkosten aus der Prozedur *kellner*, addiert 10% für Trinkgeld und druckt den Betrag der Gesamtrechnung aus.

Aufbau

Mit diesem Programm wird die Parameterübergabe auf relativ komplexe Weise dargestellt. Wir werden das Programm Schritt für Schritt erläutern, bevor wie es zusammensetzen.

```
100 REMark Prozeduren
110 RESTORE 490
120 DIM posten$(3,7), preis(3) , gericht(3)
130 REMark *** Programm ***
140 LET trinkgeld = .1
150 einlesen
    -
    -
210 DEFine PROCedure einlesen
220   FOR k = 1 TO 3
230     READ posten$(k)
240     READ preis(k)
250   END FOR k
260 END DEFine
    -
    -
    -
490 DATA ''Garnelen'', 3.5, ''Hühnchen'', 2.8,
        ''Spezial'', 3.3
```

Die Namen der Menü-Posten und ihre Preise werden in die Tabellen *posten$* und *preis* gesetzt.

Als nächstes wird eine Menü-Nummer für jeden der sechs Gäste gewählt. Die Nummer jedes bestellten Gerichtes wird in der Tabelle *gericht* gespeichert.

```
160 wahl gericht
    -
270 DEFine PROCedure wahl(gericht)
280   FOR aussuchen = 1 TO 6
290     LET nummer = RND(1 TO 3)
300     LET gericht(nummer) = gericht(nummer) + 1
310   END FOR aussuchen
320 END DEFine
```

Beachten Sie, daß der formale Parameter *gericht* sowohl:

- lokal in der Prozedur Wahl ist,
- als auch eine Tabelle im Hauptprogramm darstellt.

Die sechs Werte werden an die globale Tabelle zurückgegeben, die ebenfalls den Namen Gericht trägt. Diese Werte werden dann an die Prozedur *kellner* übergeben.

```
170 kellner gericht, rechnung
    -
    -
    -
    -
330 DEFine PROCedure kellner(gericht, kosten)
340   rechnen gericht, kosten
350   koch gericht
360 END DEFine
```

Der Kellner übergibt die Information über die Nummer jedes bestellten Gerichtes an die Prozedur, *rechnen*, die die Kosten berechnet.

```
370 DEFine PROCedure rechnen(gericht,insgesamt)
380   LET ingesamt = 0
390   FOR k = 1 TO 3
400   LET rechnung = rechnung + gericht(k)*preis(k)
410 END FOR k
420 END DEFine
```

Der Kellner gibt die selben Informationen noch an den Koch, der einfach die erforderliche Nummer für jeden Menü-Posten ausdruckt.

```
430 DEFine PROCedure koch(gericht)
440   FOR c = 1 TO 3
450     PRINT ! gericht(c) ! posten$(c) !
460   END FOR c
470 END DEFine
```

Auch hier ist wieder die Tabelle, *gericht*, in der Prozedur *koch* lokal. In ihr werden die Informationen übergeben, die die Prozedur in der **PRINT**-Anweisung benutzt.

Das vollständige Programm wird nachfolgend aufgeführt.

Programm

```
100 REMark Prozeduren
110 RESTORE 490
120 DIM posten$(3,8), preis(3), gericht(3)
130 REMark *** Programm ***
140 LET trinkgeld = .1
150 einlesen
160 wahl gericht
170 kellner gericht, rechnung
180 LET rechnung = rechnung + trinkgeld * rechnung
190 PRINT ''Sie müssen insgesamt DM'' ! rechnung !
        ''bezahlen''
200 REMARK *** Definitionen der Prozeduren ***
210 DEFine PROCedure einlesen
220   FOR k = 1 TO 3
230     READ posten$(k)
240     READ preis(k)
250   END FOR k
260 END DEFine
270 DEFine PROCEdure wahl (gericht)
280   FOR aussuchen = 1 TO 6
290     LET nummer = RND(1 TO 3)
300     LET gericht(nummer) = gericht(nummer) + 1
```

```
310   END FOR aussuchen
320 END DEFine
330 DEFine PROCedure kellner(gericht, kosten)
340   rechnen gericht, kosten
350   koch gericht
360 END DEFine
370 DEFine PROCedure rechnen(gericht, insgesamt)
380   LET insgesamt = 0
390   FOR k = 1 TO 3
400     LET insgesamt = insgesamt + gericht(k) * preis(k)
410   END FOR k
420 END DEFine
430 DEFine PROCedure koch(gericht)
440   FOR c = 1 TO 3
450     PRINT ! gericht(c) ! posten$(c)
460   END FOR c
470 END DEFine
480 REMark *** Programm-Daten ***
490 DATA ''Garnelen'', 3.5, ''Hühnchen'', 2.8,
          ''Spezial'', 3.3
```

Ausgabe

Die Ausgabe hängt von der zufälligen Auswahl von Gerichten ab. Die folgende Ausgabe stellt ein Muster dar.

```
2 Garnelen
1 Hühnchen
3 Spezial
Sie müssen insgesamt DM 21.67 bezahlen
```

Kommentar

Sicherlich ist die Benutzung von Prozeduren und Parametern in einem so einfachen Programm nicht erforderlich. Denken Sie aber daran, daß jede Unteraufgabe wesentlich komplexer sein könnte. In einer derartigen Situation würde die Benutzung von Prozeduren einen modularen Aufbau des Programms mit einem Test auf jeder einzelnen Stufe ermöglichen. Mit dem obigen Beispiel werden nur die wichtigsten Schreibweisen und Beziehungen der Prozeduren dargelegt.

Gleichermaßen verdeutlicht das nächste Beispiel die Benutzung von Funktionen.

In dem vorhergehenden Beispiel haben die Prozeduren *kellner* und *rechnen* beide genau einen Wert zurückgegeben.

Schreiben Sie die Prozeduren nun als Funktionen neu und legen Sie sämtliche dadurch erforderlich werdenden Änderungen dar.

```
DEFine FuNction kellner(gericht)
  koch gericht
  RETurn rechnen(gericht)
END DEFine

DEFine FuNction rechnen(gericht)
  LET summe = 0
  FOR k = 1 TO 3
    LET summe = summe + gericht(k)*preis(k)
END FOR k
RETurn summe
END DEFine
```

Der Funktionsaufruf für *kellner* erhält ebenfalls eine andere Form.

```
LET rechnung = kellner(gericht)
```

Dieses Programm wird wie vorher ausgeführt. Allerdings sind weniger Parameter vorhanden, obwohl die Programmstruktur ähnlich ist. Dies ist darauf zurückzuführen,daß die Funktionsnamen gleichzeitig als Parameter benutzt werden, die Informationen an den Funktionsaufruf zurückgeben.

BEISPIEL Sämtliche Variablen, die als formale Parameter in Prozeduren oder Funktionen benutzt werden, sind "sicher", da sie automatisch lokal sind. Welche Variablen, die in den Prozeduren oder Funktionen benutzt werden, sind nicht lokal? Welche zusätzlichen Anweisungen sind erforderlich, um sie zu lokalen Variablen zu machen?

Programmänderungen Die Variablen *k, pick* und *nummer* sind nicht lokal. Deshalb müssen folgende Änderungen an ihnen vorgenommen werden:

LOCAL k, nummer
LOCAL aussuchen

TYPENLOSE PARAMETER

Normalen Parametern ist kein Typ zugeordnet. Möglicherweise möchten Sie, daß eine Variable, die Zahlen verarbeitet, wie eine numerische Variable aussieht, und daß eine Variable zur Verarbeitung von Strings wie eine String-Variable aussieht. Wie immer Sie Ihre Parameter jedoch schreiben, Sie haben keinen Typ. Dies wird durch folgendes Programm unterstrichen.

Programm

```
100 REMark Zahl oder Wort
110 kellner 2
120 kellner ''Hühnchen''
130 DEFine PROCedure kellner(posten)
140   PRINT ! posten !
150 END DEFine
```

Ausgabe

```
2 Hühnchen
```

Der Typ des Parameters wird erst bestimmt, wenn die Prozedur aufgerufen wird und ein aktueller Parameter "kommt".

GELTUNGS-BEREICH DER VARIABLEN

Betrachten Sie folgende Programm und ermitteln Sie, welche beiden Zahlen ausgegeben werden:

```
100 REMark Ausblick
110 LET zahl = 1
120 test
130 DEFine PROCedure test
140   LOCal zahl
150   LET zahl = 2
160   PRINT zahl
170   versuch
180 END DEFine
190 DEFine PROCedure versuch
200   PRINT zahl
210 END DEFine
```

Natürlich wird als erste Zahl die 2 ausgedruckt. Ist jedoch die Variable *zahl* in Zeile 200 global?
Die Antwort hierauf ist, daß der Wert von *zahl* in Zeile 160 in die Prozedur *versuch* übertragen wird. Eine Variable, die in einer Prozedur lokal ist, bleibt in einer zweiten Prozedur, die von der ersten aufgerufen wird, dieselbe Variable.
Wird die Prozedur *versuch* von dem Hauptprogramm aufgerufen, so bleibt auch die Variable *zahl* in dem Hauptprogramm und in der Prozedur, *versuch*, dieselbe Zahl. Diese Tatsachen mögen auf den ersten Blick seltsam erscheinen, sind jedoch logisch.

1. Die Variable *zahl* in Zeile 110 ist global.
2. Die Variable *zahl* in der Prozedur *test* ist eindeutig in der Prozedur lokal.
3. Die Variable *zahl* in der Prozedur *versuch* "gehört" zu dem Teil des Programms, der sie als letztes aufgerufen hat.

In diesem Abschnitt wurden viele Themen behandelt, da die SuperBASIC-Funktionen und Prozeduren sehr leistungsfähig sind. Allerdings sollten alldiese Möglichkeiten nicht sofort benutzt werden. Benutzen Sie die Prozeduren und Funktionen am Anfang auf einfache Art und Weise. Sie können sehr wirksam sein und ihre Leistungsfähigkeit steht Ihnen bei Bedarf jederzeit zur Verfügung.

AUFGABEN ZU KAPITEL 15

1. Sechs Mitarbeiter einer Firma sind nur unter ihrem Nachnamen bekannt. Jeder Mitarbeiter bezahlt einer bestimmten Betrag in eine Pensionskasse, der in Form eines Prozentsatzes ausgedrückt wird. Mit den folgenden Daten werden das Gehalt und die Beiträge für die Pensionskasse der sechs Mitarbeiter dargestellt.

Müller	13,800	6.25
Maier	8,700	6.00
Schmitt	10,300	6.25
Schulze	15,000	7.00
Unger	6,200	6.00
Winkler	5,100	5.75

   Schreiben Sie nun Prozeduren für folgende Zwecke:

   - Eingabe der Daten in Tabellen
   - Berechnung der tatsächlichen Beiträge für die Pensionskassen
   - Ausgabe der Namenslisten und der berechneten Beiträge.

   Verbinden Sie die Prozeduren mit einem Hauptprogramm, das die Prozeduren zweckentsprechend aufruft.

2. Schreiben Sie eine Funktion *wahl* mit zwei Argumenten *bereich* und *fehlbetrag*. Die Funktion soll eine zufällige ganze Zahl in dem angegebenen *bereich* zurückgeben, wobei es sich jedoch nicht um den Wert von *fehlbetrag* handeln darf. Benutzen Sie die Funktion in einem Programm, das eine zufällige **PAPER**-Farbe auswählt und dann zufällige Kreise in zufälligen **INK**-Farben zeichnet, wobei keine dieser Farben der Farbe von **PAPER** entspricht.

3. Schreiben Sie die Lösung für Übung 1 neu, bei der eine Funktion *pension* Gehalt und Beiträg als Parameter übernimmt und den berechneten Beitrag für die Pensionskasse zurückgibt. Schreiben Sie mit der Funktion *pension* zwei Prozeduren, eine für die Eingabe der Daten und eine für die Ausgabe der erforderlichen Information.

4. Schreiben Sie folgendes:
   - eine Prozedur, mit der ein "Kartenpaket" eingelesen wird.
   - Eine Prozedur, mit der die Karten gemischt werden.
   - Eine Funktion, die eine Zahl als Parameter nimmt und einen String-Wert zur Beschreibung der Karte zurückgibt.
   - Eine Prozedur, die vier Spieler mit jeweils fünf Karten verarbeitet und anzeigt.
   - Ein Hauptprogramm, das die obigen Prozeduren aufruft.
     (Ein ähnliches Problem wird in Kapitel 16 erläutert.)

KAPITEL 16
VERSCHIEDENE TECHNIKEN

In diesem letzten Abschnitt stellen wir einige Anwendungen von schon besprochenen Konzepten und Möglichkeiten vor und zeigen, wie einige weitergehende Ideen angewandt werden können.

SIMULATION VON KARTENSPIELEN

Spielkarten können problemlos gespeichert und verarbeitet werden, indem sie mit den Zahlen 1 bis 52 dargestellt werden. Auf diese Weise kann jede Zahl in eine entsprechende Karte umgewandelt werden. Angenommen, die Nummer 29 wird angezeigt. Hier können Sie nun folgende Entscheidung treffen:

Karten 1 bis 13 gleich Herz
Karten 14 bis 26 gleich Kreuz
Karten 27 bis 39 gleich Karo
Karten 40 bis 52 gleich Pik

Somit wissen Sie, daß 29 "Karo" bedeutet. Den QL können Sie mit folgendem Befehl auf diese Weise programmieren:

```
LET farbe = (karte - 1) DIV 13
```

Dadurch wird ein Wert zwischen 0 und 3 berechnet, mit dem die entsprechende Farbe ausgedruckt werden kann. Der Wert kann auf den Bereich von 1 bis 13 beschränkt werden, indem folgende Anweisungen geschrieben werden:

```
LET wert = karte MOD 13
IF wert = 0 THEN LET wert = 13
```

Programm

Die Zahlen 1 bis 13 können als As, 2, 3 . . . , Bube, Dame, König oder gegebenenfalls Begriffe wie "Herz Zwei" ausgedruckt werden. Mit dem folgenden Programm wird der Name der Karte ausgedruckt, die der eingegebenen Zahl entspricht.

```
100 REMark Spielkarten
110 DIM farbe$(3,5),karte$(13,5)
120 einlesen
130 REPeat karten
140  INPUT "Geben Sie eine Kartennummer 1 - 52 ein :" !
     karte
150  IF karte<1 OR karte>52 THEN EXIT karten
160  LET farbe = (karte-1) DIV 13
170  LET wert = karte MOD 13
180  IF wert = 0 THEN LET wert = 13
190  PRINT farbe$(farbe) ! karte$(wert)
200 END REPeat karten
210 DEFine PROCedure einlesen
220  FOR f = 0 TO 3 : READ farbe$(f)
230  FOR k = 1 TO 13 : READ karte$(k)
240 END DEFine
250 DATA "Herz", "Kreuz", "Karo", "Pik"
260 DATA "Ass","Zwei", "Drei", "Vier", "Fünf",
        "Sechs", "Sieben"
270 DATA "Acht", "Neun", "Zehn", "Bube",
        "Dame", "König"
```

Ein- und Ausgabe

```
2
Herz Zwei
2
Herz Zwei
49
Pik Zehn
```

Kommentar

Beachten Sie bitte die Anwendung der **DATA**-Anweisungen. Dadurch wird praktisch eine Datei definiert, in der Daten bereitgehalten werden, die das Programm bei jeder Ausführung benötigt. Die anderen Daten, die sich bei jeder Ausführung des Pro-

gramms ändern, werden mit **INPUT**-Anweisungen eingegeben. Sind die Eingabedaten vor der Ausführung des Programms bekannt, so könnten auch andere **READ**-Anweisungen und weitere **DATA**-Anweisungen benutzt werden.

SEQUENTIELLE DATEIEN

Numerische Datei

Mit dem folgenden Programm wird eine Datei mit 100 Zahlen erstellt.

```
100 REMark Zahlen - Datei
110 OPEN_NEW #6,mdv1_zahlen
120 FOR zahlen = 1 TO 100
130   PRINT #6,zahlen
140 END FOR zahlen
150 CLOSE #6
```

Nach der Ausführung des Programms wird durch Eingabe des folgenden Befehls geprüft, ob der Dateiname "Zahlen" in dem Inhaltsverzeichnis steht:

```
DIR mdv1_zahlen
```

Sie können sich den Inhalt der Datei ohne großen Aufwand auf den Bildschirm ausgeben lassen. Geben Sie zu diesem Zweck ein:

```
COPY mdv1_zahlen TO scr
```

Dadurch wird der gesamte Inhalt der Datei auf den Bildschirm kopiert.

Sie können auch das folgende Programm benutzen, um die Datei zu lesen und den Inhalt der Datei auf dem Bildschirm anzuzeigen:

```
100 REMark Datei lesen
110 OPEN_IN #6,mdv1_zahlen
120 FOR zahl = 1 TO 100
130   INPUT #6, posten
140   PRINT ! posten !
150 END FOR zahl
160 CLOSE #6
```

Gegebenenfalls kann das Programm geändert werden, um eine Ausgabe in einer anderen Form zu erhalten.

Zeichendatei

Auf ähnliche Weise erstellen die folgenden Programme eine Datei mit 100 zufällig ausgewählten Buchstaben und lesen sie wieder zurück.

```
100 REMark Buchstaben_Datei
110 OPEN_NEW #6, mdv1_zeichen_datei
120 FOR zahl = 1 TO 100
130   LET zeich$ = CHR$(RND(65 TO 90))
140   PRINT #6, zeich$
150 END FOR zahl
160 CLOSE #6
```

```
100 REMark Buchstaben holen
110 OPEN_IN #6,mdv1_zeichen_datei
120 FOR zahl = 1 TO 100
130   INPUT #6, posten$
140   PRINT ! posten$ !
150 END FOR zahl
160 CLOSE #6
```

ERSTELLEN EINER DATENDATEI

Angenommen, Sie möchten eine einfache Datei mit Namen und Telefonnummern erstellen.

HANS	678462
PETER	896487
PAUL	249386
SEPP	584621
MAX	482349
MARIA	438975
GABI	982387

Dies ist mit dem folgenden Programm möglich.

```
100 REMark Telefon-Nummern
110 OPEN_NEW #6, mdv1_telefon
120 REPeat liste
130   INPUT name$,nummer$
140 IF name$ = ''ßß'' THEN EXIT liste
150   PRINT #6; name$ ; nummer$
160 END REPeat liste
170 CLOSE #6
```

Geben Sie **RUN** ein, gefolgt von **ENTER**. Danach verlangt das Programm von Ihnen die Eingabe von Namen und Telefonnummern. Denken Sie daran, nach jeder Eingabe eines Namens bzw. einer Telefonnummer die **ENTER**-Taste zu betätigen. Wenn Sie keine weiteren Namen eingeben wollen, dann müssen Sie anstatt eines Namens ßß eingeben.

Beachten Sie, daß die Daten "gepuffert" werden. Sie werden intern gespeichert, bis soviele Daten angesammelt sind, daß es sich lohnt, einen Stapel mit Daten an den Microdrive zu übertragen. In diesem Beispiel wird nur einmal auf das Microdrive zugegriffen, wie Sie sehen und hören können. Die Anweisung **CLOSE #6** leert den Puffer, so daß die Daten auch wirklich an die Datei übertragen werden.

KOPIEREN EINER DATEI

Nachdem eine Datei erstellt wurde, sollten Sie unmittelbar eine Sicherungskopie anfertigen. Hierzu wird folgender Befehl eingegeben:

```
COPY mdv1_telefon TO mdv2_telefon
```

LESEN EINER DATEI

Sie können sich überzeugen, daß die Datei in der richtigen Form gespeichert wurde. Also lesen Sie sie von einem Microdrive und zeigen sie auf dem Bildschirm an. Hierzu benutzen Sie folgenden Befehl:

```
COPY mdv2_telefon TO scr
```

Bei der Ausgabe auf dem Bildschirm sind nicht automatisch Leerzeichen zwischen dem Namen und der Telefonnummer vorgesehen. Allerdings kommt es am Ende jedes Eintrags zu einem Zeilenvorschub. Die Ausgabe sieht folgendermaßen aus:

```
HANS678462
PETER896487
PAUL249386
SEPP584621
MAX482349
MARIA438975
GABI982387
```

Das folgende Programm bietet eine Grundlage, die Daten übersichtlicher darzustellen.

```
100 REMark Lesen der Telefon - Nummern
110 OPEN_IN #5, mdv1_telefon
120 REPeat liste
125 IF EOF( #5 ) THEN EXIT liste
130   INPUT #5,list$
140   PRINT , list$
150 END REPeat liste
160 CLOSE #5
```

Die Daten werden wie vorher ausgedruckt. Diesmal steht jedoch jedes Paar von Name und Telefonnummer in einer Variablen *list$*, bevor es auf dem Bildschirm ausgedruckt wird. Nun haben Sie die Möglichkeit, die Ausgabe in die gewünschte Form zu bringen.

Beachten Sie bitte, daß mehr als eine String-Variable beim Erstellen der Datei mit **INPUT** und **PRINT** benutzt werden kann. Der ganze auf diese Weise erstellte Eintrag kann jedoch aus der Microdrive-Datei auf eine einzige String-Variable (im obigen Beispiel list$) eingelesen werden.

Bei dem Beispiel können Sie auch sehen, wie die Funktion **EOF** benutzt wird. In der **REPeat**-Schleife wird solange gelesen, bis die Datei zu Ende ist. Ohne Zeile 125 würde das Programm enden mit der Fehlermeldung: **IN ZEILE 130 DATEIENDE.**

Durch Zeile 125 erreichen Sie, daß das Programm am Dateiende bei einer von Ihnen gewünschten Stelle fortgesetzt wird und nicht mit einer Fehlermeldung abbricht.

SORTIERT EINFÜGEN

In der niedrigauflösenden Betriebsart stehen folgende Farben zur Verfügung (in Reihenfolge der Codenummern 0 bis 7).

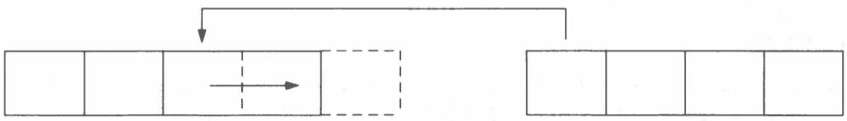
Schwarz Blau Rot Magenta Grün Cyan Gelb Weiß

BEISPIEL

Schreiben Sie ein Programm, mit dem die Farben beim *Einfügen* in alphabetischer Reihenfolge sortiert werden.

Methode

Wir schreiben die acht Farben in eine Tabelle namens *farbe$*, die wir in Gedanken in zwei Teile unterteilen:



Sortierter Teil Unsortierter Teil

Nun nehmen wir den am weitesten links stehenden Posten des unsortierten Teils und vergleichen ihn von rechts nach links mit jedem Posten in dem sortierten Teil, bis wir die richtige Stelle gefunden haben. Während des Vergleichs verschieben wir die sortierten Posten nach rechts, so daß wir den Posten sofort einfügen können, wenn wir die richtige Stelle gefunden haben, ohne weiter verschieben zu müssen.

Angenommen, wir haben bereits die ersten vier Posten sortiert. Nun konzentrieren wir uns auf Grün, dem Posten in dem unsortierten Teil, der am weitesten links steht.

1	2	3	4	▌	5	6	7	8
Schwarz	Blau	Magenta	Rot	▌	Grün	Zyan	Gelb	Weiß
	Unsortierter Teil			↑		Sortierter Teil		

1. Nun besetzen wir die Variable *vergl$* mit “Grün” und setzen eine Variable, *p*, auf 5.
2. Mit der Variablen *p* wird eventuell angegeben, wo Grün stehen soll. Wissen wir, daß Grün nach links verschoben werden muß, so wird der Wert von *p* vermindert.
3. Nun wird Grün mit Rot verglichen. Ist Grün größer als (näher an Z) oder gleich Rot, so beenden wir das Programm. Grün bleibt dann an seiner alten Stelle.

   Ansonsten kopieren wir Rot in Position 5 und vermindern den Wert von *p* um eins:

1	2	3	4	▌	5	6	7	8
Schwarz	Blau	Magenta	Rot	▌	Rot	Zyan	Gelb	Weiß
			↑					

4. Nun wiederholen wir das Verfahren. Diesmal vergleichen wir jedoch Grün mit Magenta und erhalten folgendes Ergebnis:

1	2	3	4	5	▌	6	7	8
Schwarz	Blau	Magenta	Rot	Cyan	▌		Gelb	Weiß
		↑						

5. Schließlich gehen wir wieder nach links und vergleichen Grün mit Blau. Diesmal braucht nichts verschoben oder geändert zu werden. Wir beenden die Schleife und setzen Grün in Position 3. Nun können wir uns mit dem sechsten Posten, Cyan, befassen.

1	2	3	4	5	▌	6	7	8
Schwarz	Blau	Grün	Magenta	Rot	▌	Cyan	Gelb	Weiß
						↑		

PROBLEMANALYSE

1. Als erstes speichern wir die Variable *farbe$* in einer Tabelle namens *farbe$(8)*. Weiter benutzen wir folgende Variablen:

vergl$ die gerade verglichene Farbe
p zeigt auf die Position, bei der wir testen, ob sie für die Farbe in *vergl$* die richtige ist.

2. Eine **FOR**-Schleife konzentriert sich nacheinander auf die Positionen 2 bis 8 (ein Posten ist schon sortiert).

Positionen 2 bis 8 (ein Posten ist schon sortiert).

3. Mit einer **REPeat**-Schleife werden Vergleiche vorgenommen, bis festliegt, wo der Wert von *vergl$* stehen soll.

REPeat vergleich
  IF *vergl$* nicht weiter links
  EXIT
  Eine Farbe in die Position rechts davon kopieren
  und *p* um 1 vermindern
END REPeat vergleich

4. Nachdem die **REPeat**-Schleife mit **EXIT** beendet wurde, wird die Farbe in *vergl$* in Position *p* geschrieben und die **FOR**-Schleife wird fortgesetzt.

Programmentwurf

1. Deklaration der Tabelle farbe$
2. Einlesen der Farben in die Tabelle
3. **FOR** posten = 2 **TO** 8
     LET p = posten
     LET vergl$ = farbe$(p)
       REPEAT vergleich
       IF vergl$ > = farbe$(p–1) : **EXIT** vergleich
       LETfarbe$(p) = farbe$(p–1)
       LETp = p–1
     END REPeat vergleich
     LET farbe$(p) = vergl$
   END FOR posten
4. **PRINT** sortierte Tabelle farbe$
5. **DATA**

Durch weitere Überlegungen wird ein Fehler entdeckt. Er wird problemlos ersichtlich, wenn wir Daten eingeben, bei denen der erste Posten nicht von Anfang an in der richtigen Position steht. Dann wird vom Programm *farbe$(o)* getestet – ein Tabellenelement, das wir bisher nicht besetzt haben.

Dies ist ein wohlbekanntes Problem beim Arbeiten mit Computern. Es wird am besten dadurch gelöst, indem das Ende der Tabelle gekennzeichnet wird. Direkt vor Eintritt in die **REPeat**-Schleife wird folgender Befehl benötigt:

```
LET farbe$(0) = vergl$
```

Glücklicherweise läßt SuperBASIC Null-Indizes zu, ansonsten hätte das Problem auf Kosten der Lesbarkeit gelöst werden müssen.

MODIFIZIERTES PROGRAMM

```
100 REMark Einsortieren
110 DIM farbe$(8,7)
120 FOR posten = 1 TO 8 : READ farbe$(posten)
130 FOR posten = 2 TO 8
140   LET p = posten
150   LET vergl$ = farbe$(p)
160   LET farbe$(0) = vergl$
170   REPeat vergleich
180     IF vergl$ >= farbe$(p - 1) : EXIT vergleich
190     LET farbe$(p) = farbe$(p - 1)
200     LET p = p - 1
210   END REPeat vergleich
220   LET farbe$(p) = vergl$
230 END FOR posten
240 PRINT ''Sortiert...'' : PRINT farbe$(1 TO 8)
250 DATA ''Schwarz'', ''Blau'', ''Magenta'', ''Rot''
260 DATA ''Grün'', ''Cyan'', ''Gelb'', ''Weiß''
```

Kommentar

1. Das Programm wird einwandfrei ausgeführt. Es wurde mit willkürlichen Daten getestet:

   A A A A A A A
   B A B A B A B
   A B A B A B A
   B C D E F G H
   G F E D C B A

2. Dieses Einsortieren ist nicht besonders schnell, kann jedoch beim Hinzufügen weniger Posten zu einer schon sortierten Liste von Nutzen sein. Gelegentlich ist es besser, in regelmäßigen Abständen etwas Zeit zu investieren, um die einzelnen Posten in der richtigen Reihenfolge zu halten, anstatt in großen Zeitabständen viel Zeit für ein vollständiges Neusortieren zu investieren.

Damit verfügen Sie nun über genügend Hintergrundwissen, um die Verarbeitung der Datei mit sieben Namen und Telefonnummern verfolgen zu können.

SORTIEREN EINER MCIRODRIVE-DATEI

Um die Datei namens "Telefon" in alphabetischer Reihenfolge der Namen zu sortieren, muß diese Datei in eine interne Tabelle eingelesen, sortiert und dann eine neue Datei erstellt werden, bei der die Namen in alphabetischer Reihenfolge sortiert sind.

Es wird unbedingt empfohlen, eine Datei erst zu löschen, nachdem die Ersatzdatei einwandfrei erstellt und als richtig getestet wurde. Deshalb müssen Sie die Datei zuerst aus Sicherheitsgründen unter einem anderen Namen kopieren. Hierzu wird folgendermaßen vorgegangen:

1. Kopieren Sie die Datei "Telefon" in "Telefon_temp".
2. Lesen Sie die Datei "Telefon" in eine Tabelle ein.
3. Sortieren Sie die Tabelle.
4. Pausieren Sie einen Augenblick, um zu prüfen, ob alles richtig läuft.
5. Löschen Sie die Datei "Telefon".
6. Erstellen Sie die neue Datei "Telefon".

Dies ist alles, was das Programm benötigt. Die neue Datei sollte mit folgendem Befehl geprüft werden:

```
COPY mdv1_telefon TO scr
```

Eventuell sind weitere Prüfungen sinnvoll. Danach werden folgende Befehle eingegeben:

```
DELETE mdv2_telefon
DELETE mdv1_telefon_temp
COPY mdv1_telefon TO mdv2_temp
COPY mdv1_telefon TO scr
DELETE mdv1_telefon_temp
```

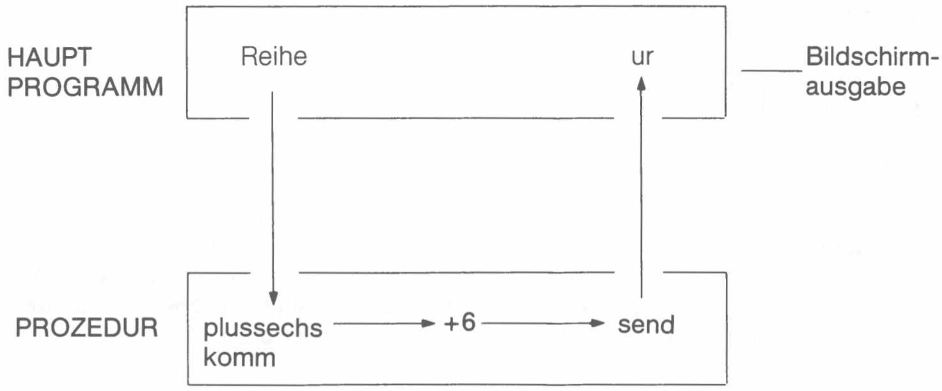
Dadurch wird die ursprünglich nicht sortierte Datei in allen Dateien durch eine sortierte Datei ersetzt.

TABELLEN-PARAMETER

In dem folgenden Programm wird die Übergabe vollständiger Tabellen zwischen einem Hauptprogramm und einer Prozedur verdeutlicht. Die Daten werden in beiden Richtungen übergeben.

In Zeile 130 wird die Tabelle namens *reihe*, in der die Zahlen 1, 2, 3 stehen, an die Prozedur *plussechs* übergeben. Der formale Parameter, *komm*, empfängt die ankommenden Daten. Die Prozedur fügt zu jedem Element sechs hinzu. In dem Tabellenparameter, *send*, stehen an dieser Stelle die Zahlen 7, 8, 9.

Diese Zahlen werden wieder an das Hauptprogramm zurückgegeben und werden zu den Tabellenwerten von *ur*. Die Werte werden ausgedruckt, um zu zeigen, daß die Daten wie erforderlich verschoben wurden.



Programm

```
100 REMark Übergabe von Tabellen
110 DIM reihe(3),ur(3)
120 FOR k = 1 TO 3 : LET reihe(k) = k
130 plussechs reihe, ur
140 FOR k = 1 TO 3 : PRINT ! ur(k) !
150 DEFine PROCedure plussechs(komm, send)
160   FOR k = 1 TO 3 : LET send(k) = komm(k) + 6
170 END DEFine
```

Ausgabe

```
7 8 9
```

Die folgende Prozedur empfängt eine Tabelle mit zu sortierenden Daten. Das Null-Element enthält die Anzahl von Posten. Beachten Sie, daß es keine Rolle spielt, ob es sich um eine numerische Tabelle oder eine String-Tabelle handelt. Mit der Datentypumwandlung werden String-Daten gegebenenfalls in numerische Daten umgewandelt.

Ein zweiter wichtiger Punkt besteht darin, daß das Tabellenelement, *komm(0)*, für zwei Zwecke benutzt wird:

- es enthält die Anzahl von zu sortierenden Posten
- in ihm wird der gerade gesetzte Posten gespeichert.

```
100 DEFine PROCedure sort(komm, send)
110   LET zahl = komm(0)
120   FOR posten = 2 TO zahl
130     LET p = posten
140     LET komm(0) = komm(p)
150     REPeat vergleich
160       IF komm(0) >= komm(p-1) THEN EXIT vergleich
170       LET komm(p) = komm(p-1)
180       LET p = p - 1
190     END REPeat vergleich
200     LET komm(p) = komm(0)
210   END FOR posten
220   FOR k = 1 TO 7 : send(k) = komm(k)
230 END DEFine
```

Mit den folgenden zusätzlichen Zeilen wird das Sortierverfahren getestet. Geben Sie als erstes **AUTO 10** ein, um die Zeilennummern ab 10 aufwärts zu starten.

```
10 REMark Sortiertest
20 DIM reihe$(7,4) , ur$(7,4)
30 LET reihe$(0) = 7
40 FOR k = 1 TO 7 : READ reihe$(k)
50 sort reihe$, ur$
60 PRINT ! ur$ !
70 DATA ''ESEL'', ''HUND'', ''HUHN'', ''MAUS'', ''EBER'',
        ''STAR'', ''IGEL''
```

Ausgabe

```
EBER ESEL HUHN HUND IGEL MAUS STAR
```

Kommentar

Mit diesem Programm wird verdeutlicht, wie einfach Tabellen bei SuperBASIC verarbeitet werden können. Sie brauchen nur die Tabellennamen zu benutzen, um sie als Parameter zu übergeben oder um die ganze Tabelle auszudrucken. Nachdem die Prozedur gesichert wurde, kann sie mit **MERGE mdv1_sort** zu einem Programm im Hauptspeicher hinzugefügt werden.

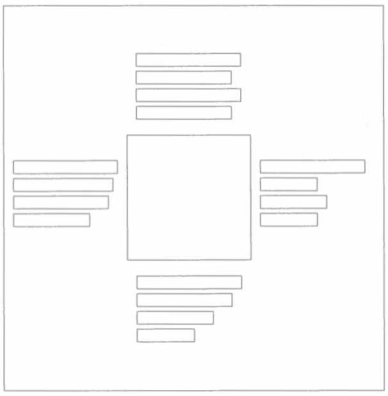
Nun verfügen wir über genügend technisches Wissen und Kenntnisse der Syntax, um einen komplexeren Bildschirmaufbau verarbeiten zu können. Angenommen, Sie möchten die Blätter von vier Kartenspielern darstellen. Ein Blatt kann beispielsweise folgendermaßen dargestellt werden:

H: A 3 7 Q
C: 5 9 J
D: 6 10 K
S: 2 4 Q

Um die Darstellung zu vereinfachen, werden Herz und Karo in Rot und Kreuz und Pik in Schwarz ausgedruckt. Eine entsprechende **STRIP**-Farbe könnte Weiß sein. Der allgemeine Hintergrund könnte Grün sein. Eine Tabelle könnte eine Farbe aufweisen, die aus zwei Farben zusammengemischt wurde.

METHODE

Da hier sehr viele Zeichen ausgedruckt werden müssen, wird die Planung am besten im Pixel-Koordinatensystem begonnen. Sie können sehen, daß sie zwölf Zeilen mit Zeichen mit etwas Abstand zwischen den Zeilen und einer gesamten Bildschirmhöhe von 256 Pixel vorsehen müssen.



Denken Sie bitte daran, daß die möglichen Zeichenhöhen 10 oder 20 Pixel betragen. Hier muß natürlich eine Höhe von 10 Pixel benutzt werden, um entsprechend Platz für den richtigen Hintergrund zu lassen.

Die Anzahl von Zeichenpositionen in einer Bildschirmzeile muß geschätzt werden. Wenn wir "10" als "Z" ausdrucken, so können sämtliche Kartenwerte als einzelnes Zeichen dargestellt werden. Außerdem gehen wir davon aus, daß wir für den Anfang maximal acht Karten derselben Farbe zulassen. Gegebenenfalls kann dieses Problem noch einmal betrachtet werden. Dadurch wären insgesamt zehn Zeichen für jedes Blatt erforderlich. In waagerechter Richtung ergeben sich also folgende Forderungen:

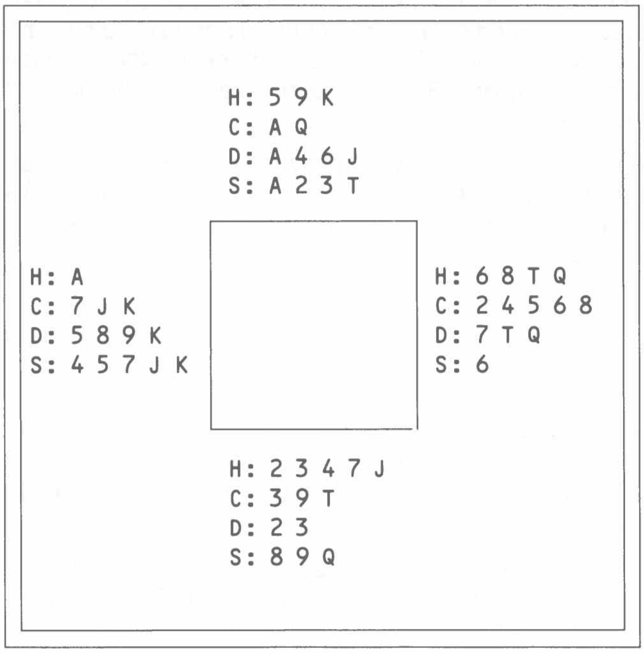
Linkes Blatt + Tabellenbreite + Rechtes Blatt

Wird zwischen den Zeichen ein Leerzeichen vorgesehen, so ergibt dies:

20 + Tabellenbreite + 20

Die Entscheidung hängt nun davon ab, welche Bildschirmbetriebsart gewählt wird. Wie wir später sehen werden, kann das Problem mit **MODE 8** gelöst werden. Wir wollen hier jedoch erst in hochauflösender Betriebsart arbeiten. Die kleinste Zeichenbreite beträgt 6 Pixel. Dies ergibt einen Gesamtwert von 240 Pixeln + Tabellenbreite. Das Diagramm ist einigermaßen ausgeglichen, wenn wir die Tabellenbreite, etwa die Hälfte von 240 Pixeln beträgt.

Deshalb wollen wir mit einer Tabellenbreite von etwa 120 Pixeln experimentieren, die später noch entsprechend angepaßt werden kann. Bei einem kleinen Test ergab sich der nachfolgend dargestellte Hintergrund.



FENSTER	440 x 220 bie 35,15 Grün mit schwarzem, 10 Pixel breitem
TABELLE	100 x 60 bei 150,60 Rot/Grünes Punktmuster
BLÄTTER	Platz für mindestens acht Karten-Symbole. Die einzelnen Blätter beginnen bei den Cursor-Positionen:

Oben	150,10
Links	260,60
Unten	150,130
Rechts	30,60

ZEICHENGRÖSSEN	Standard im 512 Modus
ZEILENHÖHE	12 Pixel
ZEICHENFARBEN	Weiß
ZEICHENSTREIFEN	Rot bei Herz und Karo Schwarz bei Kreuz und Pik

VARIABLEN

karte(52)	Speichert die Kartennummern
sort(13)	Wird für das Sortieren der Blätter benutzt
zeich$(4,2)	Speichert die Zeichen HE:, KA:, KR:, PI:
k,c,h	Arbeits-Schleifenvariablen
zuf	Zufällige Position für den Kartenaustausch
moment	Wird beim Kartenaustausch benutzt
posten	Beim Sortieren einzufügende Karte
mark	Zeiger, mit dem die Position beim Sortieren gefunden wird
vergl	Speichert die Kartennummern beim Sortieren
inc	Zeilenabstand in Pixeln
pos	Aktuelle Position
waag,senk	Cursorposition für die Zeichen
reihe	Aktuelle Reihe für die Zeichen
lin$	Erstellt eine Zeile mit Zeichen
max	Höchste Kartennummer
p	Zeigt auf die Kartenposition
n	Laufende Nummer der Karte

PROZEDUREN

mischen	Mischt 52 Karten
teilen	Teilt die Karten in vier Blätter und ruft *sortieren* auf, um jedes Blatt zu sortieren
sortieren	Sortiert 13 Karten in aufsteigender Reihenfolge
hintergrund	Liefert Hintergrundfarbe, Umrandung und Tabelle
drucken	Druckt jede Zeile mit Kartensymbolen
holt_reihe	Holt eine Reihe mit Karten und wandelt die Zahlen in die Symbole A, 2, 3, 4, 5, 6, 7, 8, 9, Z, B, D, K um.

PROGRAMM-ENTWURF

1. Deklarieren Sie die Tabellen, besetzen Sie *zeich$* mit **READ** und setzen Sie 52 Zahlen in die Tabelle *karte*.
2. Mischen Sie die Karten.
3. Teilen Sie die Karten in vier Blätter und sortieren Sie jedes Blatt.
4. Öffnen Sie das Bildschirmfenster mit **OPEN**.
5. Legen Sie den Bildschirmhintergrund fest.
6. Drucken Sie die vier Blätter aus.
7. Schließen Sie das Bildschirmfenster mit **CLOSE** ab.

```
100 DIM karte(52), sort(13), zeich$(4,2)
110 FOR k = 1 TO 4 : READ zeich$(k)
120 FOR k = 1 TO 52 : LET karte(k) = k
130 mischen
140 teilen
150 OPEN #6, scr_440 x 220a35 x 15
160 hintergrund
170 drucken
180 CLOSE #6
190 DEFine PROCedure mischen
200   FOR c = 52 TO 3 STEP -1
210     LET zuf = RND(1 TO c - 1)
220     LET moment = karte(c)
230    LET karte(c) = karte(zuf)
240     LET karte(zuf) = moment
250   END FOR c
260 END DEFine
270 DEFine PROCedure teilen
280   FOR h = 1 TO 4
290     FOR c = 1 TO 13
300       LET sort(c) = karte((h-1)*13+c)
310     END FOR c
320     sortieren
330     FOR c = 1 TO 13
340       LET karte((h-1)*13+c) = sort(c)
350     END FOR c
360   END FOR h
370 END DEFine
380 DEFine PROCedure sortieren
390   FOR posten = 2 TO 13
400     LET mark = posten
410     LET vergl = sort(mark)
420     LET sort(0) = vergl
430     REPeat vergleich
440       IF vergl >= sort(mark-1) : EXIT vergl
450       LET sort(mark) = sort(mark-1)
460       LET mark = mark - 1
470     END REPeat vergleich
480     LET sort(mark) = vergl
490   END FOR posten
500 END DEFine
510 DEFine PROCedure hintergrund
520   PAPER #6,4 : CLS #6
530   BORDER #6,10,10
540   BLOCK #6,100,60,150,60,2,4
550 END DEFine
560 DEFine PROCedure drucken
570   LET inc = 12 : INK #6,7
580   LET p = 0
590   FOR pos = 1 TO 4
600     READ waag,senk
610     FOR reihe = 1 TO 4
620       holt_reihe
630       CURSOR #6,waag,senk
640       PRINT #6,lin$
650       LET senk = senk + inc
660     END FOR reihe
670   END FOR pos
680 END DEFine
```

```
690 DEFine PROCedure holt_reihe
700   IF reihe MOD 2 = 0 THEN STRIP #6,0
710   IF reihe MOD 2 = 1 THEN STRIP #6,2
720   LET lin$ = zeich$(reihe)
730   LET max = reihe * 13
740   REPeat eine_farbe
750     LET p = p + 1
760     LET n = karte(p)
770     IF n > max THEN p = p - 1 : EXIT eine_farbe
780     LET n = n MOD 13
790     IF n = 0 THEN n = 13
800     IF n = 1 : LET sym$ = ''A''
810     IF n >= 2 AND n <= 9 : LET sym$ = n
820     IF n = 10 : LET sym$ = ''Z''
830     IF n = 11 : LET sym$ = ''B''
840     IF n = 12 : LET sym$ = ''D''
850     IF n = 13 : LET sym$ = ''K''
860     LET lin$ = lin$ & '' '' & sym$
870     IF p = 52 : EXIT eine_farbe
880     IF karte(p) > karte(p+1) : EXIT eine_farbe
890   END REPeat eine_farbe
900 END DEFine
910 DATA '' HE'', ''KA'', ''KR'', ''PI''
920 DATA 150,10,260,60,150,130,30,60
```

Kommentar

Das Programm kann auch in der achtfarbigen Betriebsart ausgeführt werden. Die vielen Zeilen mit den Kartensymbolen können jedoch die "Tabelle" überlagern, oder es kann zu einem Überlauf am Rand des Fensters kommen. Dies wird mit einer einfachen Änderung in der Prozedur *holt_reihe* von:

```
860 LET lin$ = lin$ & '' '' & sym$
```

in:

```
860 LET lin$ = lin$ & sym$
```

behoben. Die Leerzeichen zwischen den Zeichen verschwinden. Dank der größeren Zeichen können die Reihen jedoch problemlos gelesen werden. Das Programm kann also in beiden Bildschirm-Betriebsarten problemlos benutzt werden.

SCHLUSS-BEMERKUNG

Wir haben versucht, darzulegen, wie SuperBASIC zur Lösung von Problemen benutzt werden kann. Wir haben gezeigt, wie einfache Aufgaben auf einfache Art und Weise ausgeführt werden können. Sind Aufgaben in sich komplex, wie die Verarbeitung von Tabellen oder der Entwurf von Bildschirmgrafiken, so können sie mit SuperBASIC effizient mit der größtmöglichen Klarheit verarbeitet werden.

Waren Sie ein Anfänger und haben Sie sich durch einen großen Teil dieses Handbuchs durchgearbeitet, so haben Sie eine erstklassige Ausgangsposition für gutes Programmieren. Hatten Sie schon Programmier-Erfahrung, so hoffen wir, daß Sie die zusätzlichen Möglichkeiten von SuperBASIC schätzen und ausnutzen werden.

Der Bereich der Aufgaben, die mit SuperBASIC ausgeführt werden können, ist so umfangreich, daß wir in diesem Handbuch nur einen kleinen Teil davon ansprechen konnten. Wir können nicht wissen, welche der tausend Möglichkeiten für Sie von Interesse sind, hoffen jedoch, daß Sie sie nützlich, anregend und erfreulich finden.

ANTWORTEN ZUM TEST IN KAPITEL 1

1. Die **UNTERBRECHUNG** wird zur Unterbrechung eines ablaufenden Programms benutzt, weil:
   a) etwas falsch läuft und Sie es nicht verstehen
   b) das Programm nicht mehr von Interesse ist
   c) ein anderes Problem entstanden ist. (Drei Punkte)
2. Die **RÜCKSETZ**-Taste befindet sich auf der rechten Seite des Computers.
3. Die Auswirkungen der **RÜCKSETZ**-Taste können mit dem Ausschalten und Wiedereinschalten des Computers verglichen werden.
4. Die **SHIFT**-Taste:
   a) ist nur wirksam, solange Sie sie gedrückt halten, während die **FESTSTELL**-Taste wirksam bleibt, nachdem Sie sie betätigt haben; (Ein Punkt)
   b) die **SHIFT**-Taste wirkt sich auf sämtliche Buchstaben-, Ziffern- und Symboltasten aus, während sich die **FESTSTELL**-Taste nur auf Buchstaben auswirkt. (Ein Punkt)
5. Mit den Tasten **CTRL** und ← wird das Zeichen direkt links vor dem Cursor gelöscht.
6. Durch Betätigung der ⇐(**ENTER**)-Taste wird die Eingabe eines Befehls oder eines Wertes abgeschlossen und damit dem Computer zur Verarbeitung eingegeben.
7. Wir benutzen ⇐ für die **ENTER**-Taste.
8. Durch **CLS** ⇐ wird ein Teil des Bildschirms gelöscht.
9. Durch **RUN** ⇐ wird ein gespeichertes Programm ausgeführt.
10. Durch **LIST** ⇐ wird ein gespeichertes Programm auf dem Bildschirm angezeigt.
11. Durch **NEW** ⇐ wird der Hauptspeicher gelöscht, damit er zur Aufnahme eines neuen Programms bereit ist.
12. Die Befehle von SuperBASIC werden in Groß- und Kleinschreibung erkannt.
13. Der in Großbuchstaben angegebene Teil eines Befehls stellt die zulässige Abkürzung dar.

PRÜFEN SIE IHR ERGEBNIS

14 bis 16 Punkte ist sehr gut. Lesen Sie weiter.

12 oder 13 ist gut, allerdings müssen einige Teile von Kapitel 1 noch einmal gelesen werden.

10 oder 11 Punkte ist recht ordentlich, allerdings müssen einige Teile in Kapitel 1 neu gelesen und der Test noch einmal ausgeführt werden.

Weniger als 10. Sie sollten Kapitel 1 noch einmal sorgfältig durcharbeiten und den Test dann wiederholen.

ANTWORTEN ZUM TEST IN KAPITEL 2

1. Ein interner Zahlenspeicher ist wie ein Ablagefach, das Sie benennen und in das Sie Zahlen setzen können.
2. Mit einer **LET**-Anweisung, bei der ein bestimmter Name zum ersten Mal benutzt wird, wird ein Ablagefach erstellt und benannt. Zum Beispiel:

   LET zahl = 1 ⬅ (Ein Punkt)

   Eine **READ**-Anweisung, bei der ein Name zum ersten Mal benutzt wird, hat dieselbe Auswirkung. Zum Beispiel:

   READ zahl ⬅ (Ein Punkt)
3. Der Wert eines Ablagefaches kann mit einer **PRINT**-Anweisung ermittelt werden.
4. Der Fachbegriff für Ablagefach lautet 'Variable', da seine Werte bei der Ausführung des Programms variieren können.
5. Eine Variable erhält ihren ersten Wert, wenn sie das erste Mal in einer **LET-**, **INPUT-**, oder **READ**-Anweisung benutzt wird.
6. Eine Änderung im Wert einer Variablen wird im allgemeinen durch die Ausführung einer **LET**-Anweisung verursacht.
7. Das Gleichheitszeichen (=) in einer **LET**-Anweisung veranlaßt folgendes:

   Der Ausdruck auf der rechten Seite muß ausgewertet und in das auf der linken Seite angegebene Ablagefach gestellt werden, d.h. die linke Seite erhält den Wert der auf der rechten Seite angegeben ist.
8. Eine nicht numerierte Anweisung wird sofort ausgeführt.
9. Eine numerierte Anweisung wird nicht sofort ausgeführt, sondern gespeichert.
10. In einer **PRINT**-Anweisung steht der Text, der ausgedruckt werden muß, in Anführungszeichen.
11. Werden keine Anführungszeichen benutzt, so wird der Wert einer Variablen ausgedruckt.
12. Durch eine **INPUT**-Anweisung wird das Programm gestoppt, so daß Sie Daten über die Tastatur eingeben können.
13. Eine **DATA**-Anweisung wird niemals ausgeführt.
14. Sie liefert Werte für die Variablen in **READ**-Anweisungen.
15. Beispiele:

    a) tag
    b) tag_23
    c) wochen_tag (Drei Punkte)
16. Die Leertaste wird besonders dazu benutzt, vor oder hinter Befehlen Leerzeichen einzufügen, damit diese nicht als vom Benutzter ausgewählte Namen betrachtet werden.
17. Frei ausgewählte Namen sind besonders wichtig, da sie die Programme verständlicher machen. Derartige Programme sind weniger fehleranfällig und anpassungsfähiger.

PRÜFEN SIE IHR ERGEBNIS

18 bis 20 Punkte ist sehr gut. Lesen Sie weiter.

16 oder 17 ist gut, allerdings müssen einige Teile von Kapitel 2 noch einmal gelesen werden.

14 oder 15 ist recht ordentlich, allerdings müssen einige Teile von Kapitel 2 noch einmal gelesen und der Test nochmals ausgeführt werden.

Weniger als 14. Sie sollten Kapitel 2 noch einmal sorgfältig durcharbeiten und den Test wiederholen.

ANTWORTEN ZUM TEST IN KAPITEL 3

1. Ein Pixel ist der kleinest Punkt, der auf dem Bildschirm angezeigt werden kann.
2. In der niedrigauflösenden Betriebsart gibt es 256 Pixelpositionen in waagerechter Richtung.
3. In der niedrigauflösenden Betriebsart gibt es 256 Pixelpositionen in senkrechter Richtung.
4. Eine 'Adresse' wird folgendermaßen bestimmt:
   - durch den waagerechten Wert, 0 bis zu einer Zahl, die vom jeweiligen Bildschirmfenster abhängt
   - durch den senkrechten Wert, 0 bis 100
5. In der niedrigauflösenden Betriebsart stehen acht Farben zur Verfügung, einschließlich Schwarz und Weiß.
6. a) Mit **LINE** wird eine Linie gezeichnet, z. B. **LINE a, b TO x, y**
   b) Mit **INK** wird eine Farbe für das Zeichen ausgewählt, z. B. **INK 5.**
   c) Mit **PAPER** wird eine Hintergrundfarbe ausgewählt, z. B. **PAPER 7**
   d) Mit **BORDER** wird eine Umrandung gezeichnet, z. B. **BORDER 1,5**
7. **REPeat** *name* . . . **END REPeat** *name.*
8. *Eine* **REPeat**-Schleife wird beendet, wenn eine Anweisung **EXIT** *name* ausgeführt wird.
9. Die Schleifen in SuperBASIC verfügen über Namen, so daß sie auf direktem Wege mit **EXIT** verlassen werden können. Die Zeilennummern müssen nicht im voraus ermittelt werden.

PRÜFEN SIE IHR ERGEBNIS

11 oder 12 Punkte ist sehr gut. Lesen Sie weiter.

8 bis 10 ist gut, allerdings müssen einige Teile von Kapitel 3 noch einmal gelesen werden.

6 oder 7 ist recht ordentlich, allerdings müssen Teile von Kapitel 3 noch einmal gelesen und der Test erneut ausgeführt werden.

Weniger als 6. Sie sollten Kapite 3 noch einmal sorgfältig durcharbeiten und den Test wiederholen.

ANTWORTEN ZUM TEST IN KAPITEL 4

1. Ein Zeichen-String besteht aus einer Folge von Zeichen, wie beispielsweise Buchstaben, Ziffern oder anderen Symbolen.
2. Der Ausdruck "Zeichen-String" wird häufig durch "String" abgekürzt.
3. Der Name einer String-Variablen endet stets mit **$.**
4. Namen wie *wort$* werden gelegentlich als "wortdollar" ausgesprochen.
5. Mit dem Befehl **LEN** wird die Länge, d. h. die Anzahl von Zeichen in einem String ermittelt. Hat die Variable *fleisch$* beispielsweise den Wert 'Steak', so gibt die Anweisung:

   ```
   PRINT LEN(fleisch$)
   ```

   5 aus.
6. Das Symbol für die Verbindung zweier Strings lautet **&.**
7. Die Grenzen eines String können durch Anführungszeichen oder Apostrophe definiert werden.
8. Die Anführungszeichen sind nicht Bestandteil des String und werden nicht gespeichert.
9. Bei der Funktion handelt es sich um **CHR$**. Sie muß mit Klammern wie in **CHR$-(66)** oder in **CHR$(RND(65 TO 67))** benutzt werden.
10. Zufällige Buchstaben werden erzeugt mit Anweisungen wie:

    ```
    zeichencode = RND(65 TO 90)
    PRINT CHR$(zeichencode)
    ```

PRÜFEN SIE IHR ERGEBNIS

9 oder 10 Punkte ist sehr gut. Lesen Sie weiter.

7 oder 8 ist gut, allerdings müssen einige Teile von Kapitel 4 noch einmal gelesen werden.

5 oder 6 ist recht ordentlich, allerdings müssen einige Teile von Kapitel 4 noch einmal gelesen und der Test erneut ausgeführt werden.

Weniger als 5. Sie sollten Kapitel 4 noch einmal sorgfältig durcharbeiten und den Test wiederholen.

ANTWORTEN ZUM TEST IN KAPITEL 5

1. Kleinbuchstaben für Variablennamen oder Schleifennamen heben sich von Befehlen ab, die zumindest teilweise in Großbuchstaben angezeigt werden.
2. Durch die Einrückung wird deutlich, welchen Umfang und welchen Inhalt die Schleifen (und andere Strukturen) haben.
3. Namen sollten so ausgewählt werden, daß sie eine bestimmte Bedeutung haben, z.B. *zahl* oder *wort$* anstelle von *z* oder *w$*.
4. Ein gespeichertes Programm kann folgendermaßen bearbeitet werden:
   - durch Ersetzen einer Zeile
   - durch Einfügen einer Zeile
   - durch Löschen einer Zeile (Drei Punkte)
5. Die **ENTER**-Taste muß für die Eingabe eines Befehls oder einer Programmzeile benutzt werden.
6. Durch den Befehl **NEW** wird das vorhergehende SuperBASIC-Programm in dem QL gelöscht. Dadurch ist gewährleistet, daß ein neu eingegebenes Programm nicht mit einem alten Programm gemischt wird.
7. Soll eine Zeile als Teil eines Programms gespeichert werden, so muß eine Zeilennummer benutzt werden.
8. Durch den Befehl **RUN** gefolgt von ← wird ein Programm ausgeführt.
9. Mit **REMark** können Informationen in ein Programm gesetzt werden, die bei der Ausführung ignoriert werden.
10. Mit den Befehlen **SAVE** und **LOAD** können Programme auf Kassetten gespeichert und von Kassetten rückübertragen werden. (Zwei Punkte)

PRÜFEN SIE IHR ERGEBNIS

11 bis 13 Punkte ist sehr gut. Lesen Sie weiter.

9 oder 10 ist gut, allerdings müssen einige Teile von Kapitel 5 noch einmal gelesen werden.

7 oder 8 ist recht ordentlich, allerdings müssen einige Teile von Kapitel 5 noch einmal gelesen und der Test erneut ausgeführt werden.

Weniger als 7. Sie sollten Kapitel 5 noch einmal sorgfältig lesen und den Test erneut ausführen.

ANTWORTEN ZUM TEST IN KAPITEL 6

1. Es ist nicht einfach, viele verschiedene Variablennamen für die Speicherung von Daten zu finden. Können Sie genügend Namen finden, so muß jeder Name in eine **LET**- oder **READ**-Anweisung geschrieben werden, wenn keine Tabellen benutzt werden. (Zwei Punkte)
2. Eine Zahl, die als Index bezeichnet wird, ist Bestandteil eines Variablennamens einer Tabelle. Sämtliche Variablen in einer Tabelle benutzen denselben Namen, allerdings hat jede einen anderen Index.
3. Sie müssen eine Tabelle 'deklarieren', indem Sie ihre Größe (Dimension) in einer **DIM**-Anweisung angeben, die im allgemeinen am Anfang eines Programms steht, bevor die deklarierte Tabelle benutzt wird.
4. Die unterscheidende Zahl einer Tabellenvariablen wird als Index bezeichnet.
5. Die Häuser in einer Straße haben denselben Straßennamen, jedes Haus hat jedoch seine eigene Nummer.

   Betten in einer Krankenhausstation können denselben Namen der Station tragen, jedes Bett kann jedoch numeriert sein.

   Die Zellen in einem Gefängnisblock können über einen gemeinsamen Blocknamen jedoch eine andere Nummer verfügen. (Drei Punkte)
6. Eine **FOR**-Schleife wird beendet, wenn die Schleifenanweisungen mit dem letzten Wert der Schleifenvariablen ausgeführt sind.
7. Der Name einer **FOR**-Schleife ist gleichzeitig der Name der Variablen, die die Schleife steuert.
8. Die beiden Begriffe für diese Variable lauten 'Schleifenvariable' oder 'Steuervariable'.
9. Die Werte einer Schleifenvariablen können als Indizes für die Variablennamen einer Tabelle benutzt werden. Im Verlauf einer Schleife wird somit jede betroffene Tabellenvariable ein Mal benutzt.
10. Die **FOR**-Schleifen und **REPeat**-Schleifen verfügen beide über:

    a) einen öffnenden Befehl:
       REPeat, FOR

    b) eine schließende Anweisung:
       END REPeat *name,* **END FOR** *name*

    c) einen Schleifennamen.

    d) Die **FOR**-Schleife alleine verfügt über:
       eine Schleifenvariable oder Steuervariable. (Vier Punkte)

PRÜFEN SIE IHR ERGEBNIS

Dieser Test ist wesentlich anspruchsvoller als die vorhergehenden.

15 oder 16 Punkte ist ausgezeichnet. Lesen Wie weiter.

13 oder 14 ist sehr gut, denken Sie jedoch noch ein wenig über einige der Ideen nach. Betrachten Sie die einzelnen Programme, um zu sehen, wie sie arbeiten.

11 oder 12 ist gut, allerdings müssen Teile von Kapitel 6 noch einmal gelesen werden.

8 bis 10 ist recht ordentlich, allerdings müssen Teile von Kapitel 6 noch einmal gelesen und der Test erneut ausgeführt werden.

Weniger als 8. Sie sollten Kapitel 6 noch einmal sorgfältig lesen und den Test erneut ausführen.

ANTWORTEN ZUM TEST IN KAPITEL 7

1. Große oder komplexe Aufgaben werden normalerweise solange in kleinere Aufgaben unterteilt, bis sie klein genug sind, um problemlos ausgeführt werden zu können.
2. Dieses Prinzip kann auch bei der Programmierung benutzt werden, indem die gesamte Aufgabe untergliedert und für jede Teilaufgabe eine Prozedur geschrieben wird.
3. Eine einfache Prozedur
   - stellt eine eigene Gruppe von Anweisungen dar
   - ist eindeutig benannt. (Zwei Punkte)
4. Ein Prozeduraufruf bewirkt, daß:
   - die Prozedur ausgeführt wird
   - das Programm bei der Anweisung fortgesetzt wird, die dem Prozeduraufruf folgt. (Zwei Punkte)
5. Prozedurnamen können in einem Hauptprogramm benutzt werden, bevor die Prozeduren geschrieben wurden. Dadurch kann an die Aufgabe als ganzes gedacht und ein Überblick erhalten werden, ohne daß Sie sich Gedanken um Einzelheiten zu machen brauchen.
6. Wird eine Prozedurdefinition geschrieben, bevor ihr Name benutzt wird, so kann sie getestet werden. Wird sie fehlerfrei ausgeführt, so braucht man sich nicht mehr um die Einzelheiten zu kümmern. Sie brauchen sich nur ihren Namen und in groben Zügen ihre Funktion zu merken.
7. Ein Programmierer, der bis zu 30 Zeilen umfassende Programme schreiben kann, kann eine komplexe Aufgabe so in Prozeduren unterteilen, daß keine mehr als 30 Zeilen und die meisten weniger als 30 Zeilen umfassen. Auf diese Weise braucht er sich immer nur um einen Teil der Aufgabenstellung gleichzeitig zu kümmern.
8. Die Benutzung einer Prozedur würde Speicherplatz sparen, wenn sie mehr als einmal aus verschiedenen Teilen des Programms aufgerufen werden muß. Eine Prozedur muß nur einmal definiert werden, kann jedoch beliebig oft aufgerufen werden.
9. Ein Hauptprogramm kann mit **LET**- oder **READ**-Anweisungen Informationen in "Ablagefächern" seichern. Die Prozedur kann auf diese "Ablagefächer" zugreifen. Auf diese Weise benutzt die Prozedur Informationen, die ursprünglich vom Hauptprogramm erstellt wurden.

   Eine zweite Methode besteht in der Benutzung von Parametern in dem Prozeduraufruf. Diese Werte werden an Variablen in der Prozedurdefinition übergeben, die sie dann nach Bedarf benutzt. (Zwei Punkte)
10. Ein aktueller Parameter ist der aktuelle Wert, der von einem Prozeduraufruf in einem Hauptprogramm an eine Prozedur übergeben wird.
11. Ein formaler Parameter ist eine Variable in einer Porzedurdefinition, die den Wert empfängt, der der Prozedur vom Hauptprogramm übergeben wird.

PRÜFEN SIE IHR ERGEBNIS

Dies ist ein sehr anspruchsvoller Test. Möglicherweise brauchen Sie noch mehr Erfahrung bei der Benutzung von Prozeduren, beovr die Ideen voll verstanden werden. Hier handelt es sich jedoch um sehr leistungsfähige und, sobald sie einmal verstanden wurden, äußerst hilfreiche Ideen. Sie sind jeder Mühe wert.

12 bis 15 ausgezeichnet. Sie können problemlos weiterarbeiten.

10 oder 11 sehr gut. Prüfen Sie einfach bestimmte Punkte noch einmal.

8 oder 9, gut – einige Teile von Kapitel 7 müssen jedoch noch einmal gelesen werden.

6 oder 7, recht ordentlich – einige Teile von Kapitel 7 müssen jedoch noch einmal gelesen werden. Arbeiten Sie die Programme sorgfältig durch, wobei Sie sämtliche Änderungen in den Variablenwerten festhalten. Danach führen Sie den Test noch einmal aus.

Weniger als 6 – lesen Sie Kapitel 7 noch einmal durch. Arbeiten Sie sämtliche Programme sorgfältig noch einmal durch. Diese Ideen sind unter Umständen nicht einfach, sind jedoch der Mühe wert. Sobald Sie bereit sind, führen Sie den Test noch einmal aus.

sinclair

QL
Befehle

In dem Abschnitt Befehle werden sämtliche SuperBASIC-Befehle in alphabetischer Reihenfolge aufgeführt. Die Funktion des Befehls wird kurz erläutert. Darauf folgen eine kurze Definition der Syntax, sowie Beispiele für die Benutzung. Der Begriff 'Syntax' wird in dem *Abschnitt Begriffe* unter *Syntax* im einzelnen erläutert.

Bei jedem Befehl wird gegebenenfalls angegeben, auf welche Gruppe von Anweisungen er sich bezieht, d. h. **LINE** ist eine *Grafik-Anweisung.* Weitere Informationen zu diesem Befehl stehen dann beim Stichwort *Grafik* in dem *Abschnitt Begriffe.*

Gelegentlich muß mit mehr als einem Befehl gleichzeitig gearbeitet werden, z. B. mit **IF, ELSE, THEN, END IF.** Sie alle werden unter **IF** aufgeführt.

Am Ende dieses Kapitels ist ein Register enthalten, in dem angegeben wird, an welcher Stelle der SuperBASIC-Befehl beschrieben wird. So wird beispielsweise der Befehl für das Löschen des Bildschirms, **CLS** auch noch unter *Bildschirm löschen* aufgeführt.

by Stephen Berry *(Sinclair Research Limited)*

ABS
Mathematische Funktionen

ABS liefert den absoluten Wert des Parameters. Ist der Parameter positiv, so wird der Wert des Parameters zurückgegeben. Ist der Parameter negativ, so wird das Vorzeichen umgekehrt.

Syntax: ABS *(numerischer_Ausdruck)*

Beispiel:

a) `PRINT ABS(.5)`
b) `PRINT ABS(a-b)`

ACOS, ASIN ACOT, ATAN
Mathematische Funktionen

ACOS und **ASIN** berechnen den Arcuskosinus bzw. den Arcussinus. Der Betrag des Parameters muß kleiner oder gleich 1 sein. Mit **ACOT** wird der Arcuskotangens und mit **ATAN** der Arcustangens berechnet. Für die Größe des Parameters gibt es keine Einschränkung.

Die Ergebnisse dieser Funktionen sind jeweils Winkel im Bogenmaß (Radiant). Zum Beispiel bedeutet

y = ASIN (x):

Der Variablen y wird als Wert der Winkel (im Bogenmaß) zugewiesen, dessen Sinus den Wert des Arguments x hat. Entsprechendes gilt für die anderen Funktionen.

Syntax: *Winkelfunktion:= numerischer_Ausdruck* [*in Radiant*]

ACOS (Winkelfunktion) ASIN (Winkelfunktion)
ACOT (Winkelfunktion) ATAN (Winkelfunktion)

Beispiel:

a) `PRINT ATAN(tangens_des_winkels)`
b) `PRINT ASIN(1)`
c) `PRINT ACOT(3.6574)`
d) `PRINT ATAN(a-b)`

ADATE
Uhr

Mit **ADATE** kann die *Uhr* eingestellt werden.

Syntax: *Sekunden:= numerischer_Ausdruck*

ADATE *Sekunden*

Beispiel:
a) `ADATE 3600` {Stellt die Uhr um eine Stunde vor}
b) `ADATE -60` {Stellt die Uhr um eine Minute zurück}

ARC
ARC_R
Grafik

Mit **ARC** wird ein Kreisbogen zwischen zwei angegebenen Punkten in dem *Fenster* gezeichnet, das mit dem Standard-*Kanal* oder dem angegebenen *Kanal* verbunden ist. Die Endpunkte des Bogens werden in Bezug auf das *grafische Koordinatensystem* angegeben.

Mit einem einzigen **ARC**-Befehl können mehrere Bögen gezeichnet werden.

Die Endpunkte des Bogens können bezogen auf das grafische Koordinatensystem oder bezogen auf den Grafik-Cursor angegeben werden. Wird der erste Punkt nicht angegeben, so wird der Bogen von der letzten Position des Grafik-Cursors ausgehend zu dem angegebenen Punkt gezeichnet. Der gezeichnete Kreisbogen überstreicht dabei den angegebenen Winkel.

ARC zeichnet stets bezogen auf das grafische Koordinatensystem, während **ARC_R** stets bezogen auf den Grafik-Cursor zeichnet.

Syntax:
x:= *numerischer_Ausdruck*
y:= *numerischer_Ausdruck*
Winkel:= numerischer_Ausdruck (in Radiant)
Punkt:= *x,y*

Parameter_2:= | **TO** *Punkt, Winkel* 1
| *,Punkt* **TO** *Punkt, Winkel* 2

Parameter_1:= | *Punkt* **TO** *Punkt, Winkel* 1
| **TO** *Punkt, Winkel* 2

ARC [*Kanal,*] *Parameter_1* *[*Parameter_2*]*
ARC_R [*Kanal,*] *Parameter_1* *[*Parameter_2*]*

Bei 1 wird mit dem angegebenen Winkel vom ersten angegebenen Punkt zum zweiten angegebenen Punkt gezeichnet.

Bei 2 wird mit dem angegebenen Winkel von dem letzten gezeichneten Punkt zu dem angegebenen Punkt gezeichnet.

Beispiel:

a) `ARC 15, 10 TO 40,40, PI/2`
{Zeichnet einen Bogen von 15,10 bis 40,40 mit einem Winkel von π/2 Radiant}

b) `ARC TO 50,50,PI/2`
{Zeichnet einen Bogen von dem letzten gezeichneten Punkt zu 50,50 mit einem Winkel von π/2 Radiant}

c) `ARC_R 10,10 TO 55,45,.5`
{Zeichnet einen Bogen. Die Koordinaten von Anfangs- und Endpunkt beziehen sich auf den Grafik-Cursor (d.h. die zuletzt erreichte Zeichenposition). Anfangspunkt 10,10; Endpunkt 55,45; Winkel 0,5 Radiant, d.h. rund 28,6 Grad.}

Hinweis: Bei dem Befehl **ARC** *Kanal* **TO** *Parameter* darf zwischen Kanal und **TO** kein Komma und kein anderes Trennzeichen stehen.

AT
Fenster

Mit **AT** kann die Druckposition in einem gedachten Zeilen-/Spalten-Raster auf der Grundlage der aktuellen Zeichengröße geändert werden. **AT** benutzt eine modifizierte Form des *Pixel-Koordinatensystems,* wobei sich (Zeile 0, Spalte 0) in der oberen linken Ecke des Fensters befindet. **AT** bezieht sich auf die Druckposition in dem Fenster, das mit dem angegebenen Kanal bzw. dem Standardkanal verbunden ist.

Syntax: *Zeile:= numerischer_Ausdruck*
Spalte:= numerischer_Ausdruck

AT [*Kanal,*] *Zeile, Spalte*

Beispiel:

```
AT 10,20: PRINT ''Dies erscheint auf Zeile 10 ab
Spalte 20''
```

AUTO

Mit **AUTO** können Zeilennummern automatisch erzeugt werden, wenn Programme von der Tastatur in den Computer eingegeben werden. Mit **AUTO** wird die nächste Zeilennummer in der verlangten Reihenfolge erzeugt. Danach kann die nächste Programmzeile eingegeben werden. Ist die Zeile schon vorhanden, so wird eine Kopie der Zeile zusammen mit der Zeilennummer angezeigt. Wird die **ENTER**-Taste zu irgendeinem Zeitpunkt betätigt, so wird die Snytax der ganzen Zeile überprüft und die Zeile in das Programm übernommen.

AUTO wird durch Betätigung von

[CTRL] [Leertaste] beendet.

Syntax: *erste_Zeile:= Zeilennummer*
Erhöhung:= numerischer_Ausdruck

AUTO [erste_Zeile] [,Erhöhung]

Beispiel:

a) `AUTO` {Beginnt die Zeilennumerierung ab Zeile 100 und erhöht jeweils um 10}

b) `AUTO 10,5` {Beginnt die Zeilennumerierung ab Zeile 10 und erhöht jeweils um 5}

c) `AUTO ,7` {Beginnt die Zeilennumerierung ab Zeile 100 und erhöht jeweils um 7}

BAUD

Datenübertragung

Mit **BAUD** wird die Baudrate für die Datenübertragung über die beiden seriellen Einheiten festgelegt. Die Baud-Rate wird immer für beide seriellen Einheiten gemeinsam festgelegt.

Syntax: *Rate:= numerischer_Ausdruck*

BAUD *Rate*

Für den numerischen Ausdruck muß ein Wert angegeben werden, der einer der von dem QL unterstützten Baudraten entspricht:

75
300
600
1200
2400
4800
9600
19200 (nur Senden)

Wird die ausgewählte Baudrate nicht unterstützt, so wird die Fehlermeldung UNGÜLTIGER PARAMETER ausgegeben.

Beispiel:
a) `BAUD 9600`
b) `BAUD druck_geschwindigkeit`

BEEP

Akustische Signale

Mit **BEEP** werden die eingebauten Ton-Funktionen des QL aktiviert. **BEEP** kann mit einer variablen Anzahl von Parametern den erzeugten Ton gestalten. Als Mindestwerte brauchen nur die Dauer und die Tonhöhe angegeben zu werden. Wird **BEEP** ohne Parameter benutzt, so wird jeder Ton beendet.

Syntax:

Dauer:=	*numerischer_Ausdruck* {Bereich −32768 .. 32767}
Höhe:=	*numerischer_Ausdruck* {Bereich 0 .. 255}
Gradient_x:=	*numerischer_Ausdruck* {Bereich −32768 .. 32767}
Gradient_y:=	*numerischer_Ausdruck* {Bereich −8 .. 7}
Umlauf:=	*numerischer_Ausdruck* {Bereich 0 .. 15}
Verfremdung:=	*numerischer_Ausdruck* {Bereich 0 .. 15}
Zufälligkeit:=	*numerischer_Ausdruck* {Bereich 0 .. 15}

BEEP [Dauer, Höhe
[, *Höhe_2, Gradient_x, Gradient_y*
[, *Umlauf*
[, *Verfremdung*
[, *Zufälligkeit*]]]]]

Dauer	Gibt die Dauer des Tons in Einheiten von 72 µs an. Wird eine Dauer von Null angegeben, so wird der Ton gespielt, bis er durch einen anderen BEEP-Befehl beendet wird.
Höhe	Gibt die Höhe des Tons an. Eine Höhe von 1 erzeugt einen hohen und eine Höhe von 255 einen niedrigen Ton.
Höhe_2	Gibt eine zweite Höhe an, zwischen den der Ton hin- und herschwingt.
Gradient_x	Definiert die Zeit zwischen den Höhenschritten.
Gradient_y	Definiert die Größe jedes Schrittes. Mit *Grad_x* und *Grad_y* wird die Geschwindigkeit festgelegt, mit der zwischen den verschiedenen Höhen hin- und hergesprungen wird.
Umlauf	Mit diesem Parameter wird der Ton die angegebene Anzahl von Malen wiederholt. Wird für Umlauf ein Wert von 15 angegeben, so wird der Ton ständig wiederholt.
Verfremdung	Definiert eine Verfremdung für den Ton.
Zufälligkeit	Definiert Zufallseinflüsse für den Ton.

Hinweis Das Programm geht schon während der Tonerzeugung zur nächsten Anweisung über. Bei der Gestaltung einer Tonfolge sind mit dem Befehl **PAUSE** entsprechende Zwischenräume zu erzeugen.

BEEPING
Akustische Signale

BEEPING ist eine Funktion, die den Wert Null (falsch) liefert, wenn der QL gerade kein akustisches Signal abgibt. Gibt er ein akustisches Signal ab, so wird ein Wert von Eins (wahr) zurückgegeben.

Syntax: BEEPING

Beispiel:

```
100 DEFine PROCedure sei_leise
110   BEEP
120 END DEFine
130 IF BEEPING THEN sei_leise
```

BLOCK
Fenster

BLOCK füllt einen Block mit der angegebenen Größe und Form bei der angegebenen Position bezogen auf die linke obere Ecke des *Fensters* aus, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

BLOCK benutzt das *Pixel-Koordinatensystem.*

Syntax:

Breite:= *numerischer_Ausdruck*
Höhe:= *numerischer_Ausdruck*
x:= *numerischer_Ausdruck*
y:= *numerischer_Ausdruck*

BLOCK [*Kanal,*] Breite, Höhe, x, y, Farbe

Beispiel:

a)

```
BLOCK 10,10,5,5,7
```

{Ein 10 x 10 Pixel großer weißer Block bei 5,5}

b)

```
100 REMark "Balken Diagramm"
110 CSIZE 3,1
120 PRINT "Balken Diagramm"
130 LET boden = 100: breit = 20: links = 10
140 FOR balken = 1 TO 10
150   LET farbe = RND(0 TO 255)
160   LET hoch = RND(2 TO 20)
170   BLOCK breit, hoch, links+balken*breit,
        boden-hoch,0
180   BLOCK breit-2, hoch-2, links+
        balken*breit+1, boden-hoch+1,farbe
190 END FOR balken
```

{Für Fernsehgeräte `LET farbe=RND (0 to 7)` benutzen}

BORDER
Fenster

Mit **BORDER** wird ein Rahmen um das *Fenster* gesetzt, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

Bei allen nachfolgenden Operationen mit Ausnahme von **BORDER** ist die Fenstergröße um den Platz verringert, den der Rahmen beansprucht. Wird ein weiterer **BORDER**-Befehl benutzt, so wird die volle Größe des ursprünglichen Fensters vor Hinzufügung des Rahmens wiederhergestellt. Auf diese Weise verändern mehrere **BORDER**-Befehle die Größe und Farbe einer einzelnen Umrahmung. Mehrere Rahmen werden nur erstellt, wenn eine bestimmte Aktion unternommen wird (vgl. Beispiel b).

Wird **BORDER** ohne Angabe einer Farbe benutzt, so wird ein transparenter Rahmen, d. h. ein Rahmen in der Hintergrundfarbe mit der angegebenen Breite erstellt.

Syntax: *Breite:= numerischer_Ausdruck*

BORDER [*Kanal,*] *Größe* [*,Farbe*]

Beispiel:

a) `BORDER 10,0,7` {Rahmen mit schwarz/weißem Punktmuster}

b)
```
100 REMark Gespenstische Grenzen
110 FOR breit = 50 TO 2 STEP -2
120   BORDER breit, RND(0 TO 255)
130 END FOR breit
140 BORDER 50
```
{Für Fernsehgeräte wird `RND(0 TO 7)` benutzt}

CALL
Qdos

Mit dem **CALL**-Befehl kann aus SuperBASIC direkt auf gespeicherte Maschinencodeprogramme zugegriffen werden. **CALL** kann Parameter mit einer Länge von bis zu 13 Wörtern akzeptieren, die in der angegebenen Reihenfolge in die 68008 Daten- und Adreßregister (D1 bis D7, A0 bis A5) gesetzt werden.

Bei **CALL** werden keine Daten zurückgegeben.

Syntax: *Adresse:= numerischer_Ausdruck*
Daten:= numerischer_Ausdruck

CALL *Adresse,* *[*Daten*] *{maximal 13 Daten-Parameter}

Beispiel:

a) `CALL 262144,0,0,0`

b) `CALL 262500,12,3,4,1212,6`

Hinweis: Adreßregister A6 darf nicht in Routinen benutzt werden, die mit diesem Befehl aufgerufen werden. Für die Rückkehr zu SuperBASIC werden folgende Instruktionen benutzt:

```
MOVEQ #0,D0
RTS
```

CHR$
BASIC

CHR$ ist eine Funktion, die das Zeichen zurückgibt, dessen Zahlencode als Parameter angegeben wird.

CHR$ ist das Gegenstück zu CODE.

Syntax: CHR$ *(numerischer_Ausdruck)*

Beispiel:

a) `PRINT CHR$(27)` {Gibt Zeichen Nr. 27 aus; entsprechend ESCape-Taste}

b) `PRINT CHR$(65)` {Druckt A}

CIRCLE
CIRCLE_R
Grafik

Mit **CIRCLE** wird ein Kreis (oder eine Ellipse mit einem bestimmten Winkel) in einer bestimmten Position und Größe auf dem Bildschirm gezeichnet. Der Kreis wird in dem *Fenster* gezeichnet, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

Bei **CIRCLE** wird das *grafische Koordinatensystem* benutzt. Mit diesem Befehl kann bezogen auf das grafische Koordinatensystem und bezogen auf den Grafik-Cursor gezeichnet werden. Im letzteren Fall wird **CIRCLE_R** benutzt.

Mehrere Kreise oder Ellipsen können mit einem Aufruf von **CIRCLE** gezeichnet werden. Die einzelnen Parametergruppen müssen durch ein Semikolon (;) voneinander getrennt werden.

CIRCLE kann durch das Wort **ELLIPSE** ersetzt werden.

Syntax:

x:=	*numerischer_Ausdruck*
y:=	*numerischer_Ausdruck*
Radius:=	*numerischer_Ausdruck*
Exzentrizität:=	*numerischer_Ausdruck*
Winkel:=	*numerischer_Ausdruck* {*Bereich 0...2π*}

Parameter:= | *x, y* 1
| *Radius,* Exzentrizität, Winkel 2

Mit 1 wird ein Kreis gezeichnet, während mit 2 eine Ellipse mit der angegebenen Exzentrizität und dem angegebenen Winkel gezeichnet wird.

CIRCLE [*Kanal,*] Parameter *[; Parameter]*

CIRCLE_R [Kanal,] Parameter *[; Parameter]*

x	Horizontale Verschiebung des Mittelpunktes vom Ursprung des grafischen Koordinatensystems oder Grafik-Cursor.
y	Vertikale Verschiebung des Mittelpunktes vom Ursprung des grafischen Koordinatensystems oder vom Grafik-Cursor.
Radius	Radius des Kreises oder Länge der großen Hauptachse.
Exzentrizität	Verhältnis zwischen der großen und kleinen Hauptachse der Ellipse.
Winkel	Orientierung der großen Hauptachse der Ellipse bezogen auf die Bildschirm-Vertikale. Der Winkel muß im Bogenmaß angegeben werden.

Beispiel:

a) `CIRCLE 50,50,20` {Ein Kreis bei 50,50 mit dem Radius 20}

b) `CIRCLE 50,50,20,.5,0` {Eine Ellipse bei 50,50 Hauptachse 20, Exzentrizität 0.5 und auf die Vertikal-Achse ausgerichtet}

CLEAR

Mit **CLEAR** wird der SuperBASIC-Variablenbereich für das aktuelle Programm gelöscht und Platz für Qdos freigegeben.

Syntax: CLEAR

Beispiel: `CLEAR`

Kommentar

Mit **CLEAR** kann das SuperBASIC-System in einen bekannten Status zurückgesetzt werden. Wird ein Programm beispielsweise unterbrochen (oder aufgrund eines Fehlers gestoppt), während gerade eine Prozedur ausgeführt wird, so befindet sich SuperBASIC noch in der Prozedur, selbst nachdem das Programm gestoppt wurde. Mit **CLEAR** wird SuperBASIC zurückgesetzt. (Siehe **CONTINUE, RETRY.**)

CLOSE

Einheiten

Mit **CLOSE** wird der angegebene *Kanal* geschlossen. Ein mit dem Kanal verbundenes *Fenster* wird ebenfalls geschlossen.

Syntax: *Kanal:= # numerischer_Ausdruck*

CLOSE *Kanal*

Beispiel:

a) `CLOSE #4`
b) `CLOSE #eingabe_kanal`

CLS
Fenster

Mit diesem Befehl wird das *Fenster,* das mit dem angegebenen *Kanal* oder Standard-*Kanal* verbunden ist, auf die aktuelle **PAPER**-Farbe zurückgesetzt. Dies gilt nicht für einen gegebenenfalls definierten Rahmen. **CLS** akzeptiert einen wahlweisen Parameter, mit dem angegeben wird, ob nur ein Teil des Fensters gelöscht werden muß.

Syntax: *Teil:= numerischer_Ausdruck*

CLS [*Kanal,*] [*Teil*]

Teil = 0 - Ganzer Bildschirm (Standardwert, wenn kein Parameter angegeben wird)
Teil = 1 - Bildschirmanfang, ohne Cursor-Zeile
Teil = 2 - Bildschirmende, ohne Cursor-Zeile
Teil = 3 - Ganze Cursor-Zeile
Teil = 4 - Rechtes Ende der Cursor-Zeile einschließlich der Cursor-Position

Beispiel:

a) `CLS` {Das ganze Fenster}
b) `CLS 3` {Die Cursor-Zeile wird gelöscht}
c) `CLS #2,2` {Das untere Ende des zu Kanal 2 gehörigen Fensters wird gelöscht}

CODE

CODE ist eine Funktion, die den internen Code zurückgibt, mit dem das angegebene Zeichen dargestellt wird. Wird eine Zeichenfolge (String) angegeben, so gibt **CODE** die interne Darstellung des ersten Zeichens des Strings zurück.

CODE ist das Gegenstück zu **CHR$**.

Syntax: CODE *(Stringausdruck)*

Beispiel:

b) `PRINT CODE(''A'')` {Druckt 65 aus}
b) `PRINT CODE(''SuperBASIC'')` {Druckt 83 aus}

CONTINUE RETRY

Fehlerbehandlung

Mit **CONTINUE** kann ein gestopptes Programm fortgesetzt werden. Das Programm wird bei der Anweisung fortgesetzt, die auf die Anweisung folgt, bei der es gestoppt wurde. Mit **RETRY** kann eine Programmanweisung, bei der es zu einem Fehler kam, erneut ausgeführt werden.

Syntax: CONTINUE
RETRY

Beispiel:
a) `CONTINUE`
b) `RETRY`

Hinweis: Ein Programm kann nur in folgenden Fällen fortgesetzt werden:

1. Zu dem Programm wurden keine neuen Zeilen hinzugefügt.
2. Zu dem Programm wurden keine neuen Variablen hinzugefügt.
3. Keine Zeilen wurden geändert.

Der Wert der Variablen kann gesetzt oder geändert werden.

COPY COPY_N

Einheiten

Mit **COPY** wird eine Datei aus einer Eingabeeinheit in eine Ausgabeeinheit kopiert, bis eine Dateiendemarke entdeckt wird. Mit **COPY_N** wird ein eventuell vorhandener Dateikennsatz nicht kopiert. Mit diesem Befehl können Microdrive-Dateien korrekt zu einer anderen Einheitenart kopiert werden.

Einheiten, die ein Inhaltsverzeichnis benutzen, arbeiten mit Dateikennsätzen. Sie müssen mit **COPY_N** gelöscht werden, wenn in Einheiten kopiert wird, die nicht mit Inhaltsverzeichnissen arbeiten; so ist beispielsweise **mdv1** eine Einheit, die mit Inhaltsverzeichnis arbeitet, während **ser1** nicht mit Inhaltsverzeichnissen arbeitet.

Syntax: COPY *Einheit* TO *Einheit*
COPY_N *Einheit* TO *Einheit*

Die Eingabe von der Ausgangseinheit und die Ausgabe zu der Bestimmungseinheit müssen logisch möglich sein.

Beispiel:

a)	`COPY mdv1_dat_datei TO con_`	{Kopiert in das Standardfenster}
b)	`COPY neti_3 TO mdv1_daten`	{Daten werden von der Station 3 in dem Netzwerk auf mdv_daten kopiert}
c)	`COPY_N mdv1_test_daten TO ser1`	{mdv1_test_daten wird in die serielle Einheit 1 kopiert, wobei der Dateikennsatz gelöscht wird}

COS
Mathematische Funktionen

Mit **COS** wird der Kosinus des angegebenen Argumentes berechnet.

Syntax: *Winkel:= numerischer_Ausdruck*
{Bereich –10000 .. 10000 in Radiant}

COS (Winkel)

Beispiel:
a) `PRINT COS(theta)`
b) `PRINT COS(3.141593/2)`

COT
Mathematische Funktionen

Mit **COT** wird der Kotangens des angegebenen Argumentes berechnet.

Syntax: *Winkel:= numerischer Ausdruck*
{Bereich –30 000 .. 30 000 in Radiant}

COT (Winkel)

Beispiel:
a) `PRINT COT(3)`
b) `PRINT COT(3.141593/2)`

CSIZE
Fenster

Mit diesem Befehl wird eine neue Schriftgröße für das *Fenster* festgelegt, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist. Die Standardgröße beträgt 0,0 in der *Betriebsart 512* und 2,0 in der *Betriebsart 256.*

Mit 'Breite' wird die Breite des für das Zeichen vorgesehenen Platzes definiert. Mit 'Höhe' wird die Höhe des für das Zeichen vorgesehenen Platzes definiert. Die Zeichengröße wird so angepaßt, daß der zur Verfügung stehende Platz ausgefüllt wird.

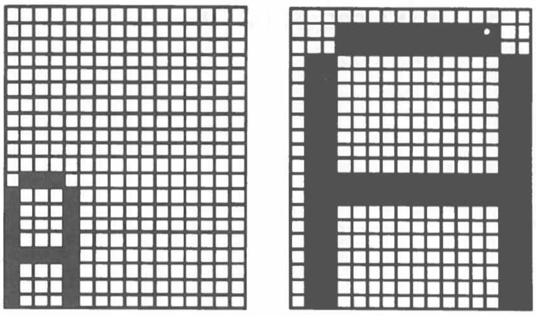


Abbildung A. Ein Zeichenquadrat

Breite	Größe	Höhe	Größe
0	6 Pixel	0	10 Pixel
1	8 Pixel	1	20 Pixel
2	12 Pixel		
3	16 Pixel		

Syntax: *Breite:= numerischer_Ausdruck* {Bereich 0 .. 3}
Höhe:= numerischer_Ausdruck {Bereich 0 .. 1}

CSIZE [*Kanal,*] *Breite, Höhe*

Beispiel:

a) `CSIZE 3,0`
b) `CSIZE 3,1`

CURSOR
Fenster

Mit dem Befehl **CURSOR** kann der Bildschirm-Cursor an eine beliebige Stelle in dem Fenster gesetzt werden, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

CURSOR benutzt das *Pixel-Koordinatensystem* relativ zum Fenster-Ursprung und definiert die Position für die obere linke Ecke des Zeichenquadrates (siehe **CSIZE**). Die Cursorgröße hängt von der gerade benutzten Zeichengröße ab.

Wird **CURSOR** mit vier Parametern benutzt, so werden das grafische und das Pixel-Koordinatensystem miteinander verbunden. Das erste Parameterpaar stellt Koordinaten in Bezug auf das grafische Koordinatensystem dar. Der dadurch definierte Punkt dient für das zweite Parameterpaar als Ursprung des Pixel-Koordinatensystems. Das zweite Paar wird als Pixel-Koordinaten, bezogen auf diesen Ursprung, interpretiert. Die Cursorposition bezeichnet immer die linke obere Ecke des Zeichenquadrates.

Auf diese Weise können Diagramme relativ einfach beschriftet werden.

Syntax: *x:= numerischer_Ausdruck*
y:= numerischer_Ausdruck

CURSOR [*Kanal,*] *x, y* [, *x,y,*]

Beispiel:

a) `CURSOR 20,30`
b) `CURSOR 50,50,10,10`

DATA READ RESTORE
BASIC

Mit den Befehlen **READ, DATA** und **RESTORE** können in einem SuperBASIC Programm enthaltene Daten bei der Ausführung Variablen zugewiesen werden.

Mit **DATA** werden die Daten markiert und definiert. **READ** greift auf die Daten zu und weist sie Variablen zu. Mit **RESTORE** können schließlich bestimmte Data-Anweisungen ausgewählt werden.

DATA: Mit diesem Befehl können Daten innerhalb eines Programms definiert werden. Die Daten können mit einer **READ**-Anweisung gelesen und Variablen zugewiesen werden. Eine **DATA**-Anweisung wird von SuperBASIC ignoriert, wenn sie während der normalen Verarbeitung angetroffen wird.

Syntax: DATA *Ausdruck* *[, *Ausdruck*]*

READ: Mit diesem Befehl werden die in **DATA**-Anweisungen stehenden Daten gelesen und einer Liste mit Variablen zugewiesen. Ursprünglich zeigt der Datenzeiger auf die erste **DATA**-Anweisung in dem Programm. Er wird nach jedem **READ**-Befehl um die Anzahl von Posten weitergesetzt, die der **READ**-Befehl verarbeitet hat. Wird das Programm erneut ausgeführt, so wird der Datenzeiger nicht zurückgesetzt. Deshalb muß ein Programm normalerweise einen expliziten **RESTORE**-Befehl enthalten.

Wird versucht, einen **READ**-Befehl auszuführen, für den kein **DATA**-Befehl vorhanden ist, so wird eine Fehlermeldung ausgegeben.

Syntax: READ *Name* *[, *Name*]*

RESTORE: Mit diesem Befehl wird der Datenzeiger, d. h. die Position, von der aus nachfolgende **READ**-Befehle ihre Daten lesen, gesetzt. Folgt auf **RESTORE** eine Zeilennummer, so wird der Datenzeiger auf diese Zeile gesetzt. Wird kein Parameter angegeben, so wird der Datenzeiger auf den Anfang des Programms zurückgesetzt.

Syntax: RESTORE [*Zeilennummer*]

Beispiel:

a)
```
100 REMark Beispiel zum 'DATA' Befehl
110 DIM wochentags$(7,4)
120 RESTORE
130 FOR zahl = 1 TO 7: READ wochentags$(zahl)
140 PRINT wochentags$: NEXT zahl
150 DATA ''MON'',''DIE'',''MIT'',''DON'',''FRE''
160 DATA ''SAM'',''SON''
```

b)
```
100 DIM monate$(12,9)
110 RESTORE
120 REMark Beispiel zum 'DATA' Befehl
130 FOR zahl = 1 TO 12: READ monate$(zahl)
140 PRINT monate$: NEXT zahl
150 DATA ''Januar'',''Februar'',''März''
160 DATA ''April'',''Mai'',''Juni''
170 DATA ''Juli'',''August'',''September''
180 DATA ''Oktober'',''November'',''Dezember''
```

Hinweis

Vor der Ausführung eines Programms wird kein implizierter **RESTORE**-Befehl ausgeführt. Dadurch kann ein einziges Programm mit verschiedenen Datengruppen ausgeführt werden. Vor der Ausführung des Programms muß entweder ein explizierter **RESTORE**- bzw. **CLEAR**-Befehl ausgeführt werden, oder es muß eine **RESTORE**-Anweisung im Programm enthalten sein.

DATE$
DATE
Uhr

DATE$ ist eine Funktion, die das Datum und die Uhrzeit in der *Uhr* des QL zurückgibt.

Der von **DATE$** zurückgegebene String hat folgendes Format:

jjjj mmm tt hh:mm:ss

wobei

jjjj	das Jahr 1984, 1985 usw.
mmm	den Monat, Jan, Feb. usw.
tt	den Tag, d.h. 01 bis 28, 29, 30, 31
hh	die Stunde, 00 bis 23
mm	die Minuten, 00 bis 59
ss	die Sekunden, 00 bis 59

angibt.

DATE gibt das Datum in Form einer Gleitkommazahl zurück, die für die Speicherung von Datums- und Zeitangaben in kompakter Form benutzt werden kann.

Wird **DATE$** mit einem numerischen Parameter benutzt, so wird der Parameter als Datumsangabe in Gleitkommaform interpretiert und in einen Datums-String umgewandelt.

Syntax:	**DATE$**	{Ermittelt die Zeit aus der Uhr}
	DATE *(numerischer_Ausdruck)*	{Ermittelt die Zeit aus dem angegebenen Parameter}
Beispiel:	a) `PRINT DATE$`	{Gibt Datum und Uhrzeit aus}
	b) `PRINT DATE$(234567)`	{Setzt 234567 in eine Datumsangabe um}

DAY$
Uhr

DAY$** ist eine Funktion, die den laufenden Wochentag zurückgibt. Wird ein Parameter angegeben, so interpretiert **DAY$ den Parameter als Datumsangabe und gibt den entsprechenden Wochentag zurück.

Syntax:	**DAY$**	{Ermittelt den Tag aus der Uhr}
	DAY$ *(numerischer_Ausdruck)*	{Ermittelt den Tag aus dem angegebenen Parameter}
Beispiel:	a) `PRINT DAY$`	{Gibt den Tag aus}
	b) `PRINT DAY$(234567)`	{Gibt den mit 234567 (Sekunden) dargestellten Tag aus}

DEFine FuNction END DEFine

Funktionen und Prozeduren

Mit **DEFine FuNction** wird eine SuperBASIC-Funktion definiert. Die Folge von Anweisungen zwischen der **DEFine FuNction** und **END DEFine**-Anweisung stellt die Funktion dar. Die Funktionsdefinition kann auch eine Liste mit *formalen Parametern* umfassen, die Daten für die Funktion liefern. Sowohl die formalen, als auch die *aktuellen Parameter* müssen in Klammern stehen. Sind für die Funktion keine Parameter erforderlich, so brauchen keine leeren Klammern angegeben zu werden.

Typ und Eigenschaften der *formalen Parameter* werden anhand der entsprechenden *aktuellen Parameter* bestimmt. Der von der Funktion zurückgegebene Datentyp wird von dem Typkennzeichen angegeben, das an den Funktionsnamen angehängt ist. Der mit der **RETURN**-Anweisung zurückgegebene Datentyp muß mit diesem Typ übereinstimmen.

Der Funktionswert wird zurückgegeben, indem ein Ausdruck an eine **RETurn**-Anweisung angehängt wird. Der Datentyp dieses Ausdrucks wird in den Datentyp der Funktion umgewandelt.

Eine Funktion wird aufgerufen, indem ihr Name in einem SuperBASIC-*Ausdruck* aufgenommen wird.

Funktionsaufrufe in SuperBASIC können rekursiv sein; d. h. eine Funktion kann sich selbst direkt oder indirekt über eine Folge anderer Aufrufe aufrufen.

Syntax:

formale_Parameter:= (*Ausdruck* *[, *Ausdruck*]*)
aktuelle_Parameter:= (*Ausdruck* *[, *Ausdruck*]*)

Typkennzeichen:= | $
| %
|

DEF FuNction *Name Typkennzeichen* [*formale_Parameter*]
  [**LOCal** *Name* *[, *Name*]*]
  Anweisungen
  RETurn *Ausdruck*
END DEFine

RETurn kann an jeder beliebigen Stelle innerhalb der Funktion stehen. **LOCal**-Anweisungen müssen vor der ersten ausführbaren Anweisung in der Funktion stehen. (Siehe DEFine PROCedure.)

Beispiel:

```
100 DEFine FuNction durchschnitt(a, b, c)
110   LOCal antwort
120   LET antwort = (a + b + c)/3
130   RETurn antwort
140 END DEFine
150 PRINT durchschnitt(1,2,3)
```

Kommentar

Um die Programme leserlicher zu gestalten, kann der Name der Funktion an die **END DEFine**-Anweisung angehängt werden. Der Name wird jedoch nicht von SuperBASIC überprüft.

DEFine PROCedure END DEFine

Funktionen und Prozeduren

Mit **DEFine PROCedure** wird eine SuperBASIC-Prozedur definiert. Die Folge von Anweisungen zwischen der **DEFine PROCedure**-Anweisung und der **END DEFine**-Anweisung stellt die Prozedur dar. Die Prozedurdefinition kann auch eine Liste mit *formalen Parametern* enthalten, die Daten für die Prozedur liefern oder aus ihr heraustragen. Bei der Prozedurdefinition müssen die *formalen Parameter* in Klammern stehen. Beim Aufruf der Prozedur wird die Liste der aktuellen Parameter jedoch nicht in Klammern gesetzt. Erfordert die Prozedur keine Parameter, so brauchen keine leeren Klammern in der Prozedurdefinition angegeben zu werden.

Typ und Eigenschaften der formalen Parameter werden anhand der entsprechenden *aktuellen Parameter* bestimmt.

Variablen können in einer Prozedur als lokale Variable definiert werden. Diese lokalen Variablen haben keine Auswirkung auf Variablen mit demselben Namen außerhalb der Prozedur. Falls erforderlich müssen lokale Tabellen innerhalb der **LOCal**-Anweisung dimensioniert werden.

Die Prozedur wird aufgerufen, indem ihr Name als erstes Element in einer SuperBASIC-Anweisung zusammen mit einer Liste von aktuellen Parametern eingegeben wird. Die Prozeduraufrufe in SuperBASIC sind rekursiv; d. h. eine Prozedur kann sich selbst direkt oder indirekt über eine Folge anderer Aufrufe aufrufen.

Eine Prozedurdefinition kann bei SuperBASIC als Befehlsdefinition betrachtet werden. Viele der Systembefehle sind selbst als Prozeduren definiert.

Syntax: *formale_Parameter:* = (*Ausdruck* *[, *Ausdruck*]*)
aktuelle_Parameter: = *Ausdruck* *[, *Ausdruck*]*

DEFine PROCedure *Name* [*formale_Parameter*]
[**LOCal** *Name* *[, *Name*]*]
Anweisungen
RETurn
END DEFine

RETurn kann an jeder beliebigen Stelle innerhalb der Prozedur stehen. **LOCal**-Anweisungen müssen, wenn vorhanden, vor der ersten ausführbaren Anweisung in der Prozedur stehen. Die **END DEFine**Anweisung wirkt wie ein automatischer Rücksprung.

Beispiel:

a)
```
100 DEFine PROCedure bildschirm_anfang
110   WINDOW 100,100,10,10
120   PAPER 7: INKO: CLS
130   BORDER 4,255
140   PRINT ''Guten Tag Allerseits''
150 END DEFine
160 bildschirm_anfang
```

b)
```
100 DEFine PROCedure abrollen (abroll_grenze)
110   LOCal zahl
120   FOR zahl = 1 TO abroll_grenze
130     SCROLL 2
140   END FOR zahl
150 END DEFine
160 abrollen 20
```

Kommentar

Um die Programme leserlicher zu gestalten, kann der Name der Prozedur an die **END DEFine**-Anweisung angehängt werden. Der Name wird jedoch nicht von SuperBASIC überprüft.

DEG
Mathematische Funktionen

DEG ist eine Funktion, die einen im Bogenmaß (Radiant) gegebenen Winkel in einen in Grad ausgedrückten Winkel umsetzt.

Syntax: **DEG** (*numerischer_Ausdruck*)

Beispiel: `PRINT DEG(PI/2)` {Druckt 90 aus}

DELETE
Microdrives

DELETE löscht eine Datei aus dem Inhaltsverzeichnis der Kassette in dem angegebenen Microdrive.

Syntax: **DELETE** *Einheit*

Mit der Einheitenangabe muß eine Microdrive-Einheit angegeben werden.

Beispiel:
a) `DELETE mdv1_alte_daten`
b) `DELETE mdv1_brief_datei`

DIM

Tabellen

Dieser Befehl definiert eine Tabelle gegenüber SuperBASIC. Mit diesem Befehl können Tabellen von *Strings, ganzen Zahlen* und *Gleitkomma-Zahlen* definiert werden. String-Tabellen enthalten Strings fester Länge, wobei der letzte *Index* als String-Länge betrachtet wird.

Die Tabellenindizes gehen von 0 bis zu dem höchsten in der **DIM**-Anweisung angegebenen Index. Auf diese Weise erstellt **DIM** eine Tabelle die in jeder Dimension ein Element mehr umfaßt, als tatsächlich angegeben wird. Eine Ausnahme bilden String-Tabellen. Bei ihnen geht der letzte Index von 1 bis zu dem höchsten Wert, der in der DIM-Anweisung an letzter Stelle steht.

Eine numerische Tabelle wird mit Null und eine String-Tabelle mit Strings der Länge Null immer dann vorbesetzt, wenn die **DIM**-Anweisung ausgeführt wird.

Syntax: *Index:= numerischer_Ausdruck*
Tabelle:= Name (Index *[, *Index*]*)

DIM *Tabelle* *[, *Tabelle*]*

Beispiel:

a) `DIM string_tabelle$(10,10,50)`
b) `DIM matrix(100,100)`

DIMN

Tabellen

DIMN ist eine Funktion, die die Maximalgröße der angegebenen Dimension einer bestimmten Tabelle zurückgibt. Wird keine Dimension angegeben, so wird die erste Dimension aufgenommen. Ist die angegebene Dimension nicht vorhanden oder wird mit dem Namen keine Tabelle angegeben, so wird Null zurückgegeben.

Syntax: *Tabelle:= Name*
Index:= numerischer_Ausdruck {1 für Dimension 1 usw.}

DIMN (*Tabelle* [, *Dimension*])

Beispiel: Hier wird die mit **DIM a (2,3,4)** definierte Tabelle betrachtet:

a) `PRINT DIMN(a,1)` {Druckt 2 aus}
b) `PRINT DIMN(a,2)` {Druckt 3 aus}
c) `PRINT DIMN(a,3)` {Druckt 4 aus}
d) `PRINT DIMN(a)` {Druckt 2 aus}
e) `PRINT DIMN(a,4)` {Druckt 0 aus}

DIR
Microdrives

Mit **DIR** wird das Inhaltsverzeichnis der Kassette in dem angegebenen Microdrive ermittelt und in dem *Fenster* angezeigt, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

Syntax: **DIR** *Einheit*

Mit der Einheitenangabe muß eine gültige *Microdrive Einheit* angegeben werden.

Das von **DIR** ausgegebene Verzeichnis hat folgendes Format:

Freie_Sektoren: =	*Anzahl freier Sektoren*
Verfügbare_Sektoren: =	*Höchstzahl von Sektoren auf dieser Kassette*
Dateiname: =	*Ein Dateiname*
Bildschirmformat:	*Name des Datenträgers* *Freie_Sektoren / verfügbare Sektoren* **Sektoren** *Dateiname* – *Dateiname*

Beispiel:

a) `DIR mdv1_`
b) `DIR "mdv2_"`
c) `DIR "mdv" & microdrive_nummer$ & "_"`

Bildschirmformat:

```
BASIC
183/221 Sektoren
demo_1
demo_1_alt
demo_2
```

DIV
Operator

DIV ist der Operator für die ganzzahlige Division.

Syntax: *numerischer_Ausdruck* **DIV** *numerischer_Ausdruck*

Beispiel:

a) `PRINT 5 DIV 2` {Gibt 2 aus}
b) `PRINT -5 DIV 2` {Gibt –3 aus}

DLINE
BASIC

Mit **DLINE** wird eine einzelne Zeile oder ein Bereich von Zeilen aus einem SuperBASIC-Programm gelöscht.

Syntax:	*Bereich:=*	| *Zeilennummer* **TO** *Zeilennummer*	1
		| *Zeilennummer* **TO**	2
		| **TO** *Zeilennummer*	3
		| *Zeilennummer*	4

DLINE *Bereich* *[, Bereich]*

1 – Löscht einen Bereich von Zeilen
2 – Löscht ab der angegebenen Zeile bis zum Ende
3 – Löscht ab dem Beginn bis zur angegebenen Zeile
4 – Löscht die angegebene Zeile

Beispiel:

a) `DLINE 10 TO 70, 80, 200 TO 400`
{Löscht die Zeilen 10 bis 70 einschließlich, sowie Zeile 80 und die Zeilen 200 bis 400 einschließlich}

b) `DLINE`
{Löscht nichts}

EDIT

Mit dem **EDIT**-Befehl wird der SuperBASIC Zeileneditor aufgerufen.

Der **EDIT**-Befehl ist eng mit dem **AUTO**-Befehl verwandt, wobei der einzige Unterschied in den Standardwerten liegt. Bei **EDIT** werden die Zeilen standardmäßig um Null erhöht. Auf diese Weise wird eine einzige Zeile bearbeitet, wenn nicht mit einem zweiten Parameter eine fortlaufende *Erhöhung* der Zeilennummer angegeben wird.

Ist die angegebene Zeile schon vorhanden, so wird die Zeile angezeigt und kann bearbeitet werden. Ist die Zeile nicht vorhanden, so wird die Zeilennummer angezeigt und die Zeile kann eingegeben werden.

Der Cursor kann innerhalb der zu bearbeitenden Zeile mit dem QL-Cursorsteuertasten bewegt werden.

[→]	Cursor nach rechts	
[←]	Cursor nach links	
[↑]	Cursor nach oben	Wie **ENTER**, zeigt jedoch automatisch die vorhergehende Zeile als nächste zu bearbeitende Zeile an.
[↓]	Cursor nach unten	Wie **ENTER**, zeigt jedoch automatisch die nächste Zeile als nächste zu bearbeitende Zeile an.
[CTRL] [→]		Löscht das Zeichen, auf dem der Cursor steht.
[CTRL] [←]		Löscht das links vom Cursor stehende Zeichen.

Sobald die Zeile die richtige Form aufweist, wird sie durch Betätigung der **ENTER**-Taste in das Programm eingegeben.

Wurde eine *Erhöhung* angegeben, so wird die nächste Zeile in der durch die Erhöhung gegebenen Reihenfolge bearbeitet. Ansonsten wird die Bearbeitung beendet.

Syntax: *Erhöhung:= numerischer_Ausdruck*

EDIT *Zeilennummer* [,*Erhöhung*]

Beispiel:

a) `EDIT 10` {Bearbeitet nur Zeile 10}
b) `EDIT 20,10` {Bearbeitet die Zeilen 20, 30, usw.}

EOF
Einheiten

EOF ist eine Funktion, mit der festgestellt wird, ob eine Dateiende-Bedingung bei einem bestimmten Kanal erreicht ist. Wird **EOF** ohne eine Kanalangabe benutzt, so wird mit **EOF** festgestellt, ob alle Posten der DATA-Anweisung innerhalb des Programms gelesen wurden.

Syntax: EOF [(*Kanal*)]

Beispiel:
a) `IF EOF(#6) THEN STOP`
b) `IF EOF THEN PRINT "KEINE DATEN MEHR"`

EXEC
EXEC_W
Qdos

Mit **EXEC** und **EXEC_W** wird eine Folge von Programmen geladen, wobei die Programme dann parallel ausgeführt werden.

Mit **EXEC** wird zu dem Befehlsprozessor zurückgegangen, nachdem die Ausführung sämtlicher Programme begonnen wurde, während bei **EXEC_W** gewartet wird, bis das Programm beendet ist, bevor zum Befehlsprozessor zurückgegangen wird.

Syntax: *Programm:=* Einheit {Wird benutzt, um eine Microdrive-Datei anzugeben, die das Programm enthält}

EXEC *Programm*

Beispiel:
a) `EXEC mdv1_kommunikationen`
b) `EXEC_W mdv1_druck_prozess`

EXIT
Wiederholung

EXIT führt zum Verlassen einer Schleife und setzt die Verarbeitung nach dem ENDE der angegebenen **FOR**- oder **REPeat**-Struktur fort.

Syntax: **EXIT** *Name*

Beispiel:

a)
```
100 REMark schleife
110 LET zahl = 0
120 REPeat schleife
130   LET zahl = zahl + 1
140   PRINT zahl
150   IF zahl = 20 THEN EXIT schleife
160 END REPeat schleife
```
{Die Schleife wird beendet, wenn die Variable Zahl einen Wert von 20 aufweist}

b)
```
100 FOR n = 1 TO 1000
110   REMark Programm Befehle
120   REMark Programm Befehle
130   IF RND > .5 THEN EXIT n
140 END FOR n
```
{Die Schleife wird beendet, wenn eine größere Zufallszahl als 0.5 erzeugt wird}

EXP
Mathematische Funktion

EXP gibt den Wert von e hoch dem angegebenen Parameter zurück.

Syntax: **EXP**(*numerischer_Ausdruck*) {Bereich –500 .. 500}

Beispiel:

a) `PRINT EXP(3)`

b) `PRINT EXP(3.141593)`

FILL
Grafik

Mit **FILL** wird das *Ausfüllen von Grafiken* ein- oder ausgeschaltet. **FILL** füllt jede Figur aus, die mit den *Grafik-* oder *Turtle Grafik* Prozeduren gezeichnet wird. Es werden nur konvexe Figuren korrekt ausgefüllt. (D. h. Figuren, deren Begrenzung nicht in das Innere der Figur hineinragt.) Nichtkonvexe Figuren müssen in konvexe Teile zerlegt werden. Die Teilfiguren müssen dann einzeln gefüllt werden.

Nachdem das Ausfüllen beendet ist, muß **FILL 0** aufgerufen werden.

Syntax: *Schalter:= numerischer_Ausdruck* {*Bereich 0... 1*}

FILL [*Kanal,*] *Schalter*

Ist der Wert von Schalter weder Null noch Eins dann ist das Ergebnis unvorhersehbar.

Beispiel:

a) `FILL 1:LINE 10,10 TO 50,50 TO 30,90 TO 10,10: FILL 0`
{Zeichnet ein ausgefülltes Dreieck}

b) `FILL 1: CIRCLE 50,50,20: FILL 0`
{Zeichnet einen ausgefüllten Kreis}

FILL$
Stringtabellen

FILL$ ist eine Funktion, die einen String mit einer angegebenen Länge zurückgibt, der aus einer Wiederholung von einem oder zwei Zeichen besteht.

Syntax: **FILL$** (*String_Ausdruck, numerischer_Ausdruck*)

Der für **FILL$** angegebene String_Ausdruck muß eine Länge von einem oder zwei Zeichen aufweisen. Ist er länger, werden die letzten Zeichen ignoriert.

Beispiel:

a) `PRINT FILL$(''a'',5)` {Druckt aaaaa aus}
b) `PRINT FILL$(''oO'',7)` {Druckt oOoOoOo aus}

FLASH
Fenster

Mit **FLASH** wird das Blinken von Schrift auf dem Bildschirm ein- und ausgeschaltet. **FLASH** ist nur in der *niedrigauflösenden Betriebsart* wirksam.
FLASH ist in dem *Fenster* wirksam, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

Syntax: *Schalter:= ganzzahliger_Ausdruck*{Bereich 0..1}
FLASH [*Kanal,*] *Schalter*
Schalter = 0 schaltet das Blinken aus.
Schalter = 1 schaltet das Blinken ein.

Beispiel:

```
100 PRINT "Ein ";
110 FLASH 1
120 PRINT "blinkendes ";
130 FLASH 0
140 PRINT "Wort"
```

Hinweis

Wird ein Teil eines blinkenden Zeichens überschrieben, so kann es zu unvorhergesehenen Effekten kommen. Dies muß unbedingt vermieden werden.

FOR
END FOR
Wiederholung

Mit der **FOR**-Anweisung kann eine Gruppe von SuperBASIC-Anweisungen eine bestimmte Anzahl von Malen wiederholt werden. Die **FOR**-Anweisung kann sowohl in einer langen als auch einer kurzen Form benutzt werden.
NEXT und **END FOR** können zusammen in derselben **FOR**-Schleife benutzt werden, um einen *Schleifennachsatz* zu erzeugen, d.h. eine Gruppe von SuperBASIC-Anweisungen, die nicht ausgeführt werden, wenn eine Schleife mit einer **EXIT**-Anweisung beendet wird. Wenn die **FOR**-Schleife normal beendet wird, so werden sie hingegen ausgeführt.

Definition: *for_Element:=* | *numerischer_Ausdruck*
| *numerischer_Ausdr* **TO** *numerischer_Ausdr*
| *numerischer_Ausdr* **TO** *numerischer_Ausdr* **STEP** *numerischer_Ausdr*

for_Liste:= for_Element *[, *for_Element*]*

Kurze Form

Auf die **FOR**-Anweisung folgt in derselben Zeile eine Reihe von SuperBASIC-Anweisungen. Diese Anweisungen werden dann wiederholt ausgeführt. Die Schleife wird so oft durchlaufen, wie in der **FOR**-Anweisung angegeben. Danach wird die Verarbeitung in der nächsten Zeile fortgesetzt. Die **FOR**-Anweisung muß nicht mit **NEXT** oder **END FOR** beendet werden. **FOR**-Schleifen in einer einzigen Zeile dürfen nicht verschachtelt sein.

Syntax: **FOR** *Variable = for_Liste: Anweisung* *[: *Anweisung*]*

Beispiel:

a) `FOR i = 1,2,3,4 TO 7 STEP 2 : PRINT i`

b) `FOR element = erstes TO letztes : LET puffer(element) = 0`

Lange Form

Die **FOR**-Anweisung ist die letzte Anweisung in einer Zeile. Nachfolgende Zeilen enthalten eine Reihe von SuperBASIC-Anweisungen, die mit einer **END FOR** Anweisung beendet werden. Die Anweisungen zwischen der **FOR**-Anweisung und der **END FOR** Anweisung werden so viele Male nacheinander verarbeitet, wie in der **FOR**-Anweisung angegeben.

Syntax: **FOR** *Variable = for_Liste*
Anweisungen
END FOR *Variable*

Beispiel:

```
100 INPUT "Daten bitte" ! x
110 LET produkt = 1
120 FOR wert = x TO 1 STEP -1
130   LET produkt = produkt * wert
140   PRINT x !!!! produkt
150   IF produkt > 1E20 THEN
160     PRINT "Eine sehr große Zahl"
170     EXIT wert
180   END IF
190 END FOR wert
```

Hinweis

Für eine **FOR**-Schleife muß als Zählvariable eine Gleitkommavariable benutzt werden.

FORMAT
Microdrives

Mit **FORMAT** wird die Kassette in dem angegebenen Microdrive formatiert und zur Benutzung vorbereitet.

Syntax: FORMAT [Kanal,] *Einheit*

Mit "Einheit" wird das für die Formatierung zu benutzende Microdrive angegeben. Die Kassette erhält den in der Einheitenbezeichnung genannten Namenszusatz als Datenträgername. Die Anzahl von fehlerfreien Sektoren und die Gesamtanzahl der zur Verfügung stehenden Sektoren wird in den Standard- oder den angegebenen Kanal geschrieben.

Neue Kassetten sollten unbedingt mehrmals vor der Benutzung formatiert werden. Dadurch wird die Oberfläche des Bandes vorbereitet und das Band erhält eine größere Kapazität.

Beispiel: a) `FORMAT mdv1_daten_kass`
b) `FORMAT mdv2_prv_briefe`

Hinweis

Mit **FORMAT** kann eine benutzte Kassette erneut formatiert werden. Allerdings gehen in diesem Fall sämtliche auf der Kassette stehenden Daten verloren.

GOSUB

Im Hinblick auf die Kompatibilität mit anderen BASIC-Dialekten enthält SuperBASIC die **GOSUB**-Anweisung. Nach **GOSUB** wird die Verarbeitung bei der angegebenen Zeilennummer fortgesetzt, die **GOSUB** spezifiziert.

Die Zeilennummer kann in Form eines Ausdrucks angegeben werden.

Syntax: GOSUB *Zeilennummer*

Beispiel: a) `GOSUB 100`
b) `GOSUB 4*variable`

Kommentar

In SuperBASIC ist die **GOSUB**-Anweisung eigentlich überflüssig, denn es besitzt weitaus leistungsfähigere Möglichkeiten wie z. B. Prozeduren.

GOTO

Im Hinblick auf die Kompatibilität mit anderen BASIC-Dialekten enthält SuperBASIC die **GOTO**-Anweisung. Nach **GOTO** wird die Verarbeitung bei der angegebenen Zeilennummer fortgesetzt. Die Zeilennummer kann in Form eines Ausdruckes angegeben werden.

Syntax: GOTO *Zeilennummer*

Beispiel:
a) `GOTO programm_anfang`
b) `GOTO 9999`

Kommentar Die **GOTO**-Anweisung ist in SuperBASIC eigentlich überflüssig, denn es besitzt erheblich leistungsfähigere Möglichkeiten, den Programmablauf zu steuern.

IF THEN ELSE END IF

Mit der **IF**-Anweisung können Bedingungen getestet werden. Das Ergebnis dieses Tests legt dann den nachfolgenden Programmablauf fest.

Die **IF**-Anweisung kann sowohl in einer langen als auch in einer kurzen Form benutzt werden:

Kurze Form Auf den Befehl **THEN** folgt in derselben Zeile eine Reihe von SuperBASIC-Anweisungen. Diese Reihe von SuperBASIC-Anweisungen kann auch einen **ELSE**-Befehl enthalten. Ist der Ausdruck in der **IF**-Anweisung wahr, so werden die Anweisungen zwischen **THEN**- und **ELSE** verarbeitet. Ist die Bedingung falsch, so werden die Anweisungen zwischen **ELSE** und dem Ende der Zeile verarbeitet.

Enthält die Folge von SuperBASIC-Anweisungen keinen **ELSE**-Befehl und ist der Ausdruck in der **IF**-Anweisung wahr, so werden die Anweisungen zwischen dem **THEN**-Befehl und dem Ende der Zeile verarbeitet. Ist der Ausdruck falsch, so wird die Verarbeitung in der nächsten Zeile fortgesetzt.

Syntax: *Anweisungen:= Anweisung* *[: *Anweisung*]*

IF *Ausdruck* THEN *Anweisungen* [:Else *Anweisungen*]

Beispiel:
a) `IF a=32 THEN PRINT ''Außerhalb der Grenze'': ELSE PRINT ''O.K.''`
b) `IF test > maximum THEN LET maximum = test`
c) `IF ''1''+1=2 THEN PRINT ''Datentypumwandlung O.K.''`

Lange Form 1 Der **THEN**-Befehl ist die letzte Anweisung der logischen Zeile. Im Anschluß an die **IF**-Anweisungen steht eine Folge von SuperBASIC-Anweisungen. Die Folge wird mit der **END IF**-Anweisung beendet. Die Folge von SuperBASIC-Anweisungen wird ausgeführt, wenn der Ausdruck in der **IF**-Anweisung wahr ist. Der **ELSE**-Befehl und die zweite Folge SuperBASIC-Anweisungen können entfallen.

Lange Form 2 Der **THEN**-Befehl ist die letzte Anweisung auf der logischen Zeile. In den nächsten Zeilen steht eine Reihe von SuperBASIC-Anweisungen, die mit dem **ELSE**-Befehl beendet wird. Ist die in der **IF**-Anweisung enthaltene Bedingung wahr, so wird diese erste Folge von SuperBASIC-Anweisungen ausgeführt. Im Anschluß an den **ELSE**-Befehl wird eine zweite Folge mit SuperBASIC-Anweisungen eingegeben. Sie wird mit dem Befehl **END IF** beendet. Ist die in der **IF**-Anweisung enthaltene Bedingung falsch, so wird diese zweite Folge der SuperBASIC-Anweisungen verarbeitet.

Ist die Bedingung ein arithmetischer Ausdruck, so ist sie falsch, wenn der Ausdruck den Wert Null ergibt. Andernfalls ist die Bedingung wahr.

Syntax: IF *Ausdruck* THEN
Anweisungen
[ELSE *Anweisungen*] END IF

Beispiel:

```
100 LET grenze = 10
110 INPUT "Schreiben Sie eine Zahl" ! zahl
120 IF zahl > grenze THEN
130   PRINT "Außerhalb der Grenze"
140 ELSE
150   PRINT "Innerhalb der Grenze"
160 END IF
```

Kommentar

In allen drei Formen der **IF**-Anweisung kann der **THEN**-Befehl entfallen. In der kurzen Form muß er durch einen Doppelpunkt ersetzt werden, um das Ende des **IF**-Befehls vom Anfang der nächsten Anweisung zu unterscheiden. In der langen Form kann **THEN** vollständig weggelassen werden.

Verschachtelung

IF-Anweisungen können so tief verschachtelt werden, wie der Benutzer dies für erforderlich hält. (Dies hängt natürlich auch vom zur Verfügung stehenden Speicherplatz ab.) Allerdings kann es zu Verwirrungen führen, wenn festgelegt werden muß, welcher **ELSE**, **END IF** usw. Befehl zu **IF** gehört. SuperBASIC ordnet verschachtelte **ELSE**-Anweisungen usw. der am nächsten stehenden **IF**-Anweisung zu. Zum Beispiel:

Beispiel:

```
100 IF a = b THEN
110   IF c = d THEN
120     PRINT "Fehler"
130   ELSE
140     PRINT "Kein Fehler"
150   END IF
160 ELSE
170   PRINT "Nicht überprüft worden"
180 END IF
```

Der **ELSE**-Befehl in Zeile 130 gehört zu dem zweiten **IF-Befehl. Der ELSE**-Befehl in Zeile 160 gehört zu dem ersten **IF**-Befehl (in Zeile 100).

INK

Fenster

Mit diesem Befehl wird die aktuelle Schriftfarbe festgelegt, d.h. die Farbe, mit der die Ausgabe geschrieben wird. **INK** ist für das *Fenster* wirksam, das mit dem angegebenen *Kanal* oder Standard-*Kanal* verbunden ist.

Syntax: INK [*Kanal,*] *Farbe*

Beispiel:

a) `INK 5`
b) `INK 6,2`
c) `INK #2,255`

INKEY$

INKEY$ ist eine Funktion, die ein einzelnes Zeichen zurückgibt, das von dem angegebenen *Kanal* oder dem Standard-*Kanal* eingegeben wurde.

Bei Bedarf kann angegeben werden, ob das Programm auf die Eingabe eine bestimmte Zeit warten soll. Wird kein Parameter angegeben, so wartet das Programm nicht.

Syntax: **INKEY$** [|(*Kanal*)
|(*Kanal, Zeit*)
|(*Zeit*)]

Zeit = 1 .. 32767 {Hier wird während der angegebenen Anzahl von Zeiteinheiten (fünfzigstel Sekunden) gewartet}
Zeit = –1 {Hier wird solange gewartet, bis die Eingabe erfolgt}
Zeit = 0 {Das Programm wartet nicht}.

Beispiel:

a) `PRINT INKEY$` {Eingabe aus dem Standardkanal}
b) `PRINT INKEY$(#4)` {Eingabe aus Kanal 4}
c) `PRINT INKEY$(50)` {Es wird eine Sekunde (50 Zeiteinheiten) lang gewartet, danach wird das Zeichen in jedem Fall zurückgegeben}
d) `PRINT INKEY$(0)` {Tastaturabfrage ohne Wartezeit}
e) `PRINT INKEY$(#3,100)` {Es wird zwei Sekunden (100 Zeiteinheiten) lang auf eine Eingabe von Kanal 3 gewartet}

Hinweis Erfolgt während der Wartezeit keine Eingabe, so liefert **INKEY$** einen String der Länge Null.

INPUT

Mit **INPUT** können Daten vom Benutzer direkt über die QL-Tastatur in ein SuperBASIC-Programm eingegeben werden.
SuperBASIC hält das Programm an, bis alle angeforderten Daten eingegeben sind. Danach wird das Programm fortgesetzt. Jede einzelne Dateneingabe muß durch Betätigung der **ENTER**-Taste beendet werden.

Mit **INPUT** werden Daten von dem angegebenen *Kanal* oder dem Standard-*Kanal* eingegeben.

Wird eine Eingabe von einem bestimmten Konsolkanal gefordert, so blinkt der Cursor in dem mit diesem Kanal verbundenen Fenster.

Syntax: *Trennzeichen* = | !
| ,
| \
| ;
| TO

Bedienerhinweis: = [*Kanal,*] *Ausdruck Trennzeichen*

INPUT [*Bedienerhinweis*] [*Kanal*] Variable *[, *Variable*]*

Beispiel:

a)
```
INPUT ("Letzter Tip " & tip & "Neuer Tip ?") !-
  tip
```
b)
```
INPUT "Geben Sie Ihren Tip"; tip
```
c)
```
100 INPUT "Tabellen-Größe ?" ! grenze
110 DIM tabelle(grenze-1)
120 FOR element = 0 TO grenze-1
130   INPUT ("DATEN für Element " & element) !
        tabelle(element)
140 END FOR element
150 PRINT tabelle
```

Hinweis: Der Ausdruck im Bedienerhinweis muß aus Konstanten bestehen. Beachten Sie, daß eine Variable dadurch zur Konstanten wird, daß man sie in Klammern setzt (s. Beispiel a).

INSTR
Operator

INSTR ist ein Operator, mit dem festgestellt wird, ob ein bestimmter Teilstring in dem angegebenen String enthalten ist. Wird der Teilstring gefunden, so wird seine Position zurückgegeben. Wird der Teilstring nicht gefunden, so gibt **INSTR** Null zurück.

Null kann als falsch interpretiert werden, d. h. der Teilstring war in dem angegebenen String nicht enthalten. Ein von Null abweichender Wert, d. h. die Position des Teilstrings, kann als wahr interpretiert werden, d. h. der Teilstring war in dem angegebenen String enthalten.

Syntax: *String_Ausdruck* **INSTR** *String-Ausdruck*

Beispiel:

a) `PRINT "a" INSTR "Katze"` {Druckt 2 aus}

b) `PRINT "WELLE" INSTR "Schwellenangst"` {Druckt 4 aus}

c) `PRINT "x" INSTR "Eier"` {Druckt 0 aus}

INT
Mathematische Funktionen

INT gibt den ganzzahligen Teil des angegebenen Gleitkommaausdruckes zurück.

Es wird immer abgerundet.

Syntax: **INT** (*numerischer_Ausdruck*)

Beispiel:

a) `PRINT INT(x)`

b) `PRINT INT(3.141593/2)`

KEYROW

KEYROW ist eine Funktion, die den momentanen Status einer Reihe von Tasten betrachtet. (In der folgenden Tabelle wird angegeben, wie die Tasten in einer 8 x8 Matrix abgebildet werden). Für **KEYROW** muß als Parameter eine ganze Zahl im Bereich von 0 bis 7 angegeben werden. Mit dieser Zahl wird festgelegt, welche Tastenreihe betrachtet wird. **KEYROW** gibt eine ganze Zahl zwischen 0 und 255 zurück, mit der in binärer Darstellung angegeben wird, welche Tasten in der betreffenden Reihe betätigt wurden.

Da **KEYROW** als Alternative für die normale Eingabe über die Tastatur mit **INKEY$** oder **INPUT** benutzt wird, wird jedes Zeichen in dem Eingabepuffer der Tastatur durch **KEYROW** gelöscht. Tastenbetätigungen, die vor dem Aufruf von **KEYROW** vorgenommen wurden, werden demzufolge von einem nachfolgenden **INKEY$** oder **INPUT** Befehl nicht gelesen.

Hier wird darauf hingewiesen, daß es zu überraschenden Ergebnissen führen kann, wenn mehrere Tasten gleichzeitig betätigt werden. Werden drei Tasten in den Ecken eines Rechtecks in der Matrix gleichzeitig betätigt, so sieht es aus, als wäre die Taste in der vierten Ecke ebenfalls betätigt worden. Die drei Sondertasten **CTRL, SHIFT** und **ALT** sind eine Ausnahme zu dieser Regel. Sie beeinflussen die anderen Tasten nicht.

Syntax: *Zeile:= numerischer_Ausdruck*{Bereich 0..7}

KEYROW (*Zeile*)

Beispiel:

```
100 REMark Lassen Sie dieses Programm laufen und
         drücken Sie dabei auf ein paar Tasten
110 REPeat schleife
120  CURSOR 0,0
130  FOR reihe = 0 TO 7
140    PRINT reihe !!! KEYROW(reihe) ; '' ''
150  END FOR reihe
160 END REPeat schleife
```

TASTATUR-MATRIX

Zeile \ Spalte	1	2	4	8	16	32	64	128
7	SHIFT	CTRL	ALT	X	V	/	N	,
6	8	2	6	Q	E	O	T	U
5	9	W	I	TAB	R	–	Y	
4	L	3	H	1	A	P	D	J
3	l	CAPS LOCK	K	S	F	=	G	;
2	|	Z	.	C	B	£	M	~
1	ENTER	←	up	ESC	→	\	SPACE	down
0	F4	F1	5	F2	F3	F5	4	7

LBYTES
Einheiten
Microdrives

Mit **LBYTES** werden Daten aus einer Datei, beginnend bei der angegebenen Startadresse, in den Speicher geladen.

Syntax: *Startadresse:= numerischer_Ausdruck*

LBYTES *Einheit, Startadresse*

Beispiel:

a) `LBYTES mdv1_bildschirm, 131072`
{Lädt eine Bildschirm-Datei}

b) `LBYTES mdv1_programm, start_adresse`
{Lädt ein Programm bei einer angegebenen Adresse}

LEN
Stringtabellen

LEN ist eine Funktion, die die Länge des angegebenen *Stringausdrucks* zurückgibt.

Syntax: LEN (*String-Ausdruck*)

Beispiel:

a)
```
PRINT LEN("LEN wird die Länge von diesem String
     ermitteln")
```

b) `PRINT LEN(ausgabe_string$)`

LET

Mit **LET** wird eine SuperBASIC-Wertzuweisung begonnen. Die Angabe des Schlüsselwortes **LET** kann entfallen. Die Zuweisung kann sowohl für String- als auch für numerische Zuweisungen benutzt werden. SuperBASIC wandelt unvereinbare Datentypen wann immer möglich automatisch in vereinbare Datentypen um.

Syntax: [**LET**] *Variable = Ausdruck*

Beispiel:

a) `LET a = 1 + 2`
b) `LET a$ = ''12345''`
c) `LET a$ = 6789`
d) `b$ = test_daten`

LINE
LINE_R
Grafik

Mit **LINE** kann eine gerade Linie zwischen zwei Punkten in dem *Fenster* gezeichnet werden, das mit dem Standard-*Kanal* oder dem angegebenen *Kanal* verbunden ist. Die Enden der Linie werden im *Grafik-Koordinatensystem* angegeben.

Mit einem einzigen **LINE**-Befehl können mehrere Linien gezeichnet werden.

Normalerweise müssen die beiden Endpunkte für eine Linie angegeben werden. Diese Enden können entweder als Koordinaten bezogen auf den Ursprung des *grafischen Koordinatensystems* oder als Koordinaten bezogen auf den *Grafik-Cursor* angegeben werden. Wird der erste Punkt weggelassen, so wird eine Linie vom Grafik-Cursor zu dem angegebenen Punkt gezeichnet. Wird der zweite Punkt weggelassen, so wird der Grafik-Cursor verschoben, jedoch keine Linie gezeichnet.

Bei **LINE** wird stets mit Koordinaten bezogen auf den Ursprung des grafischen Koordinatensystem gezeichnet, während mit **LINE_R** stets bezogen auf den Grafik-Cursor gezeichnet wird.

Syntax:

x:= numerischer_Ausdruck
y:= numerischer_Ausdruck
Punkt:= x, y

Parameter_2:= | **TO** *Punkt* 1
| *,Punkt* **TO** *Punkt* 2

Parameter_1:= | *Punkt* **TO** *Punkt* 1
| **TO** *Punkt* 2
| *Punkt* 3

LINE [*Kanal,*] *Parameter_1* *[*Parameter_2*]*

LINE_R [*Kanal,*] *Parameter_1* *[*Parameter_2*]*

1 zeichnet von dem angegebenen Punkt bis zum nächsten angegebenen Punkt.
2 zeichnet von dem letzten gezeichneten Punkt zum angegebenen Punkt.
3 geht zu dem angegebenen Punkt – eine Linie wird nicht gezeichnet.

Beispiel:

a) `LINE 0,0 TO 0,50 TO 50,50 TO 50,0 TO 0,0` {Ein Quadrat}
b) `LINE TO .75,.5` {Eine Linie}
c) `LINE 25,25` {Verschiebt den Grafik-Cursor}

Hinweis: Bei dem Befehl **LINE** *Kanal* **TO** *Parameter_1* darf zwischen *Kanal* und **TO** kein Komma und kein anderes Trennzeichen stehen.

LIST

Mit **LIST** kann eine SuperBASIC-Zeile oder eine Gruppe von Zeilen in einem bestimmten *Kanal* oder dem Standard-*Kanal* aufgelistet werden.

LIST wird durch Betätigung von:

CTRL | Leertaste | abgebrochen.

Syntax: *Zeile:=* | *Zeilennummer* TO *Zeilennummer* 1
| *Zeilennummer* TO 2
| TO *Zeilennummer* 3
| *Zeilennummer* 4
| 5

LIST [*Kanal,*] *Zeile* *[,*Zeile*]*

1 – Listet von der ersten angegebenen Zeile bis zur zweiten angegebenen Zeile auf.
2 – Listet von der angegebenen Zeile bis zum Ende auf.
3 – Listet vom Anfang bis zur angegebenen Zeile auf.
4 – Listet die angegebene Zeile auf.
5 – Listet das ganze Programm auf.

Beispiel:
a) `LIST` {Listet alle Zeilen auf}
b) `LIST 10 TO 300` {Listet die Zeilen 10 bis 300 auf}
c) `LIST 12,20,50` {Listet nur die Zeilen 12, 20 und 50 auf}

Hinweis

Betätigen von **CTRL** und **F5** hält das Listen bis zum weiteren Betätigen von **CTRL** und **F5** an.

Kommentar

Geht die Ausgabe von **LIST** an einen *Kanal,* der mit einem Drucker verbunden ist, so wird mit **LIST** eine Liste des Programms ausgedruckt.

LOAD

Einheiten
Microdrives

LOAD lädt ein SuperBASIC-Programm von einer beliebigen QL-Einheit. Bevor ein neues *Programm* geladen wird, führt **LOAD** automatisch einen **NEW**-Befehl aus. Auf diese Weise wird jedes vorher geladene Programm von **LOAD** gelöscht.

Weist eine Zeilenangabe beim Laden eine falsche SuperBASIC Syntax auf, so wird das Wort **MISTake** zwischen der Zeilennummer und dem eigentlichen Text der Zeile eingefügt. Bei der Ausführung verursacht eine derartige Zeile eine Fehlermeldung.

Syntax: LOAD *Einheit*

Beispiel:
a) `LOAD "mdv1_test_programm"`
b) `LOAD mdv1_spiele`
c) `LOAD neti_3`
d) `LOAD ser1_e`

LOCal
Funktionen und Prozeduren

Mit **LOCal** können *Variable* als **LOCal** für eine *Funktion oder Prozedur* definiert werden. Lokale Variable sind nur innerhalb der Funktion oder Prozedur vorhanden, in der sie definiert werden, oder in Prozeduren und Funktionen, die von der Funktion oder Prozedur aufgerufen werden, in der diese lokalen Variable definiert sind. Bei Beendigung der Funktion oder Prozedur sind sie verloren. Lokale Variable sind von Variablen gleichen Namens außerhalb der definierenden Funktion oder Prozedur unabhängig. *Tabellen* können als lokale Tabellen definiert werden, indem sie innerhalb der **LOCal** Anweisung dimensioniert werden.

Die **LOCal** Anweisung muß vor der ersten ausführbaren Anweisung in der Funktion oder Prozedur stehen, in der sie benutzt wird.

Syntax: LOCal *Name* *[,*Name*]*

Beispiel:
a) `LOCal a, b, c(10,10)`
b) `LOCal temp_daten`

Kommentar

Werden Variablen als **LOCal** definiert, so können Variablennamen innerhalb von Funktionen und Prozeduren benutzt werden, ohne daß deshalb wichtige Variablen mit demselben Namen außerhalb der Funktion oder Prozedur verändert werden.

LN
LOG10
Mathematische Funktionen

LN gibt den natürlichen Logarithmus des angegebenen Argumentes zurück. Mit **LOG10** wird der Zehnerlogarithmus zurückgegeben.

Syntax:
LOG10 (*numerischer_Ausdruck*) {Bereich größer als Null}
LN (*numerischer_Ausdruck*) {Bereich größer als Null}

Beispiel:
a) `PRINT LOG10(20)`
b) `PRINT LN(3.141593)`

LRUN
Einheiten
Microdrives

Mit **LRUN** wird ein SuperBASIC-*Programm* von einer angegebenen *Einheit* geladen und ausgeführt. Bevor ein anderes Programm geladen wird, führt **LRUN** einen **NEW**-Befehl aus. Auf diese Weise wird jedes vorher gespeicherte SuperBASIC-Programm durch Ausführung von **LRUN** gelöscht.

Weist eine Zeileneingabe beim Laden eine falsche SuperBASIC-Syntax auf, so wird das Wort **MISTake** zwischen der Zeilennummer und dem eigentlichen Text der Zeile eingefügt. Bei der Ausführung verursacht eine derartige Zeile eine Fehlermeldung.

Syntax: LRUN *Einheit*

Beispiel:
a) `LRUN mdv2_text`
b) `LRUN mdv1_spiel`

MERGE
Einheiten
Microdrives

Mit **MERGE** wird eine *Datei* von der angegebenen *Einheit* als SuperBASIC-*Programm* geladen. Enthält die neue Datei eine *Zeilennummer,* die noch nicht in dem schon im QL gespeicherten Programm steht, so wird die Zeile hinzugefügt. Enthält die neue Datei Programmzeilen mit Nummern, die auch das bereits gespeicherte Programm enthält, so werden diese Zeilen des alten Programms durch die neuen Zeilen ersetzt. Alle anderen alten Programmzeilen bleiben unverändert.

Weist eine Zeileneingabe während eines **MERGE**-Befehls eine falsche SuperBASIC-Syntax auf, so wird das Wort **MISTake** zwischen der Zeilennummer und dem eigentlichen Text der Zeile eingefügt. Bei der Ausführung verursacht eine derartige Zeile eine Fehlermeldung.

Syntax: MERGE *Einheit*

Beispiel:
a) `MERGE mdv1_neue_daten`
b) `MERGE mdv1_beigemischtes_programm`

Hinweis

MERGE kann nicht in Prozeduren oder Funktionen benutzt werden.

MOD
Operatoren

MOD ist ein Operator, der den Rest der Division ganzer Zahlen zurückgibt.

Syntax: *numerischer_Ausdruck* **MOD** *numerischer_Ausdruck*

Beispiel:
a) `PRINT 5 MOD 2` {Druckt 1 aus}
b) `PRINT 5 MOD 3` {Druckt 2 aus}

MODE
Bildschirm

Mit **MODE** wird die Auflösung des Bildschirms und die Anzahl von Farben festgelegt, die auf dem Bildschirm angezeigt werden können. Mit **MODE** werden sämtliche gerade auf dem Bildschirm angezeigten *Fenster* gelöscht. Ihre Position und Farbe sowie eventuell vorhandene Ränder werden jedoch beibehalten. Wird in die niedrigauflösende Betriebsart (acht Farben) gegangen, so wird die Mindestzeichengröße auf 2,0 festgelegt.

Syntax: **MODE** *numerischer_Ausdruck*

8 oder 256 wählt die niedrigauflösende Betriebsart aus
4 oder 512 wählt die hochauflösende Betriebsart aus

Beispiel:
a) `MODE 256`
b) `MODE 4`

MOVE
Turtle-Grafik

Mit **MOVE** wird die *Turtle* über eine bestimmte Entfernung in der laufenden Richtung verschoben. Die Richtung kann mit den Befehlen **TURN** und **TURNTO** angegeben werden. Mit dem Maßstabsfaktor für die Grafik wird festgelegt, wie weit die Turtle tatsächlich verschoben wird. Wird eine negative Entfernung angegeben, so wird die Turtle zurückbewegt.

Die Turtle wird in dem *Fenster* verschoben, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

Syntax: *Entfernung:= numerischer_Ausdruck*

MOVE [*Kanal,*] *Entfernung*

Beispiel:

a) `MOVE #2,20` {Verschiebt die Turtle in Kanal 2 um 20 Einheiten nach vorne}

b) `MOVE -50` {Verschiebt die Turtle im Standardkanal um 50 Einheiten zurück}

MRUN
Einheiten
Microdrives

MRUN interpretiert eine *Datei* als ein SuperBASIC-*Programm* und mischt sie mit dem gerade geladenen Programm.

Wird **MRUN** als *direkter Befehl* benutzt, so wird das neue Programm von Anfang an ausgeführt. Wird **MRUN** als Programm-*Anweisung* benutzt, so wird die Verarbeitung mit der Zeile fortgesetzt, die auf **MRUN** folgt.

Weist eine Zeileneingabe beim Mischen eine falsche SuperBASIC-Syntax auf, so wird das Wort **MISTake** zwischen der Zeilennummer und dem eigentlichen Text der Zeile eingefügt. Bei der Ausführung verursacht eine derartige Zeile eine Fehlermeldung.

Syntax: MRUN *Einheit*

Beispiel:

a) `MRUN mdv1_ketten_programm`

b) `MRUN mdv1_neue_daten`

Hinweis

MRUN kann nicht in Prozeduren oder Funktionen benutzt werden.

NET

Netzwerk Mit **NET** kann die Nummer einer Station in dem *Netzwerk* festgelegt werden. Wird nicht ausdrücklich eine Stationsnummer angegeben, so geht der QL von Stationsnummer 1 aus.

Syntax: *Station:= numerischer_Ausdruck* {Bereich 1 .. 127}

NET *Station*

Beispiel:

a) `NET 63`

b) `NET 1`

Kommentar Wenn mehrere Stationen in dem Netzwerk dieselbe Stationsnummer besitzen, so kann es zu Verwechslungen kommen.

NEW

Mit **NEW** werden das alte *Programm,* alle *Variablen* und *Kanäle* mit Ausnahme von 0,1 und 2 gelöscht.

Syntax: NEW

Beispiel: `NEW`

NEXT
Wiederholung

Mit **NEXT** werden **REPeat**- und **FOR**-Schleifen beendet. Außerdem kann mit diesem Befehl ein *Schleifen-Nachsatz* in **REPeat**- oder **FOR**-Schleifen erstellt werden.

Syntax: **NEXT** *Name*

Der Name muß mit dem Namen der Schleife übereinstimmen, die von **NEXT** gesteuert wird.

Beispiel:

a)
```
100 REMark Diese Schleife läuft endlos
110 REPeat unendliche_schleife
120   PRINT "Noch am Laufen"
130 NEXT unendliche_schleife
```

b)
```
100 REMark Diese Schleife wird 20 mal wiederholt
110 LET grenze = 20
120 FOR zahl = 1 TO grenze
130   PRINT zahl
140 NEXT zahl
```

c)
```
100 REMark Diese Schleife wird Ihnen sagen
      wenn 30 erreicht ist
110 REPeat schleife
120   LET nummer = RND(1 TO 100)
130   IF nummer <> 30 THEN NEXT schleife
140   PRINT nummer; " ist 30"
150 EXIT schleife
160 END REPeat schleife
```

NEXT in REPeat

Wird **NEXT** innerhalb einer **REPeat – END REPeat**-Struktur benutzt, so wird die Verarbeitung bei der Anweisung fortgesetzt, die auf die entsprechende **REPeat**-Anweisung folgt.

NEXT in FOR

Die **NEXT**-Anweisung veranlaßt einen erneuten Eingang in die **FOR**-Schleife. Dabei wird die Zählvariable auf den nächsten Wert gesetzt. Wenn die Zählvariable dabei ihren Endwert erreicht oder überschreitet, so wird die Verarbeitung bei der Anweisung fortgesetzt, die auf die **NEXT**-Anweisung folgt. Im anderen Fall wird die Verarbeitung bei der Anweisung fortgesetzt, die auf die **FOR**-Anweisung folgt.

ON...GOTO
ON...GOSUB

SuperBASIC enthält die **ON GOTO**- und **ON GOSUB**-Anweisungen, damit eine Kompatibilität mit anderen BASIC-Dialekten erreicht wird. Mit diesen Anweisungen kann eine bestimmte *Programmzeile* aus einer Liste verschiedener Möglichkeiten zur weiteren Verarbeitung ausgewählt werden. Welche Zeilennummer ausgewählt wird, hängt vom Wert der Steuervariablen ab. Hat die Steuervariable beim Aufruf der Anweisung einen Wert, der größer ist, als die Anzahl der zur Auswahl stehenden Zeilennummern, dann erfolgt eine Fehlermeldung.

Syntax: **ON** *Variable* **GOTO** *Ausdruck* *[, *Ausdruck*]*
ON *Variable* **GOSUB** *Ausdruck* *[, *Ausdruck*]*

Beispiel:

a)
```
ON x GO TO 10, 20, 30, 40
```

b)
```
ON irgendeine_variable
GOSUB 1000, 2000, 3000, 4000
```

Kommentar

Diese beiden BASIC-Befehle können durch **SELect** ersetzt werden.

OPEN
OPEN_IN
OPEN_NEW

Einheiten
Microdrives

Mit **OPEN** kann ein logischer *Kanal* mit einer physikalischen QL-*Einheit* für die E/A verbunden werden.

Ist der Kanal mit einem Microdrive verbunden, so kann es sich bei der Microdrive-Datei um eine bestehende oder eine neue Datei handeln. Der Befehl **OPEN_IN** eröffnet eine schon bestehende Microdrive-Datei für die Eingabe, während **OPEN_NEW** eine neue Microdrive-Datei für die Ausgabe anlegt und eröffnet.

Snytax: *Kanal:= # numerischer_Ausdruck*

OPEN *Kanal, Einheit*

Beispiele:

a) `OPEN #5, f_name$`

b) `OPEN_IN #9, "mdv1_datei_name"`
{Öffnet die Datei mdv1_datei_name}

c) `OPEN_NEW #7, mdv1_daten_datei`
{Öffnet die Datei mdv1_daten_datei}

d `OPEN #6, con_10 x 20a20 x 20_32`
{Öffnet Kanal 6 für die Konsoleinheit, wobei eine Fenstergröße mit 10 x 20 Pixeln bei Position 20,20 und ein 32 Byte umfassenden Tastatur-Eingabepuffer erstellt wird}

e) `OPEN #8, mdv1_ein_ausgabe_datei`

OVER

Fenster

Mit **OVER** wird die Art des *Überschreibens* ausgewählt, die in dem mit dem angegebenen *Kanal* oder Standard-*Kanal* verbundenen Fenster beabsichtigt ist. Die ausgewählte Art der Überschreibung bleibt wirksam, bis **OVER** erneut benutzt wird.

Syntax: *Schalter:= numerischer_Ausdruck* {Bereich −1 .. 1}

OVER [*Kanal*,] *Schalter*

Schalter = 0 –	Druckt *Schriftfarbe* auf *Streifen*
Schalter = 1 –	Überschreibt bestehenden Text ohne etwas zu löschen.
Schalter = −1 –	Bildet bei jedem Pixel das exclusive Oder (XOR) mit dem vorhandenen und dem neu zu druckenden Pixel. Ist das Ergebnis wahr (d.h. liegt einmal Schriftfarbe und das andere mal Hintergrundfarbe vor), dann wird das Pixel in Schriftfarbe, andernfalls in Hintergrundfarbe gesetzt.

Beispiel:

a) `OVER 1` {setzt "Überschreiben"}

b)
```
100 REMark Schattenschrift
110 PAPER 7: INK 0: OVER 1: CLS
120 CSIZE 3,1
130 FOR i = 0 TO 10
140   CURSOR i,i
150   IF i = 10 THEN INK 2
160   PRINT "Schatten"
170 END FOR i
```

PAN
Fenster

Mit dem Befehl **PAN** wird ein ganzes Fenster um die angegebene Anzahl von Pixeln nach rechts oder links verschoben. **PAPER** wird in den Bildschirm gerollt, um den gelöschten Bereich auszufüllen.

PAN wirkt auf den angegebenen oder den Standard-*Kanal.*

Ein zweiter Parameter kann bei Bedarf angegeben werden, mit dem nur ein Teil des *Bildschirms* waagerecht verschoben wird.

Syntax:

Entfernung:= *numerischer_Ausdruck*
Teil:= *numerischer_Ausdruck*

PAN [,*Kanal,*] *Enfertnung* [,*Teil*]

Teil = 0 – Ganzer Bildschirm (bzw. kein Parameter)
Teil = 3 – Ganze Cursor-Zeile
Teil = 4 – Rechtes Ende der Cursor-Zeile einschließlich der Cursor-Position

Ergibt die Auswertung des Ausdrucks einen positiven Wert, so wird der Bildschirminhalt nach rechts geschoben.

Beispiel:

a) `PAN #2,50` {Verschiebt um 50 Pixel nach rechts}
b) `PAN -100` {Verschiebt um 100 Pixel nach links}
c) `PAN 50,3` {Verschiebt die ganze Cursor-Zeile um 50 Pixel nach rechts}

Hinweis

Wurde als *Farbe* des Hintergrundes ein Punktmuster gewählt oder wird der Bildschirm in niedrigauflösender Betriebsart betrieben, so bleibt das ursprüngliche Punktmuster nach **PAN** erhalten, wenn der Bildschirminhalt jeweils um das Vielfache von zwei Pixeln waagerecht verschoben wird.

PAPER
Fenster, Farbe

Mit **PAPER** wird eine neue *Hintergrundfarbe* festgelegt (d. h. die Farbe, die von **CLS, PAN, SCROLL** usw. benutzt wird). Die ausgewählte Hintergrundfarbe bleibt wirksam, bis der **PAPER**-Befehl das nächste Mal benutzt wird. Mit **PAPER** wird auch die **STRIP**-Farbe und das Punktmuster festgelegt.

PAPER ändert die Farbe in dem *Fenster,* das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

Syntax: **PAPER** [*Kanal,*] *Farbe* [,*Streifenfarbe*] [,*Punktmuster*]

Beispiel:

a) `PAPER #3,7` {Weißer Hintergrund bei Kanal 3}
b) `PAPER 7,2` {Weiß/rotes Punktmuster}
c) `PAPER 255` {Schwarz/weißes Punktmuster}
d)

```
100 REMark Zeigt Farben und Punktmuster
110 FOR farbe = 0 TO 7
120   FOR kontrast = 0 TO 7
130     FOR punktmuster = 0 TO 3
140       PAPER farbe, kontrast, punktmuster
150       SCROLL 6
160     END FOR punktmuster
170   END FOR kontrast
180 END FOR farbe
```

{Kann für Fernsehgeräte nicht benutzt werden}

PAUSE

Der **PAUSE**-Befehl veranlaßt ein *Programm,* eine bestimmte Zeit zu warten. Wartezeiten werden in Einheiten von 1/50 Sekunden angegeben. Eine Tastatureingabe beendet jede Pause. Wird kein Parameter für die Wartezeit angegeben, dann dauert die Pause solange, bis sie durch einen Tastendruck beendet wird.

Syntax: *Wartezeit:= numerischer_Ausdruck*

PAUSE [*Wartezeit*]

Beispiel:
a) `PAUSE 50` {Wartet eine Sekunde lang}
b) `PAUSE 500` {Wartet zehn Sekunden lang}

PEEK
PEEK_W
PEEK_L
BASIC

PEEK ist eine Funktion, die den Inhalt des angegebenen Speicherplatzes zurückgibt.

PEEK weist drei Formen auf, mit denen auf ein Byte (8 Bit), ein Wort (16 Bit) bzw. ein langes Wort (32 Bit) zugegriffen wird.

Syntax: *Adresse:= numerischer_Ausdruck*

PEEK(*Adresse*) {Zugriff auf ein Byte}
PEEK_W(*Adresse*) {Zugriff auf ein Wort}
PEEK_L(*Adresse*) {Zugriff auf ein langes Wort}

Beispiel:
a) `PRINT PEEK(12245)` {Byte-Inhalt von Adresse 12245}
b) `PRINT PEEK_W(12)` {Wort-Inhalt von Adresse 12 und 13}
c) `PRINT PEEK_L(1000)` {Inhalt des langen Wortes in Adresse 1000}

Hinweis Bei dem Zugriff auf ein Wort und ein langes Wort muß eine geradzahlige Adresse angegeben werden.

PENUP
PENDOWN
Turtle-Grafik

Mit diesem Befehl wird der 'Stift' in *Turtle-Grafiken* bewegt. Ist der Stift abgehoben, so wird nichts gezeichnet. Ist der Stift aufgesetzt, so werden Linien gezeichnet, während die Turtle über den Bildschirm läuft.

Die Linie wird in dem *Fenster* gezeichnet, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist. Die Linie wird in der aktuellen Schriftfarbe des Kanals gezeichnet, an den die Ausgabe geht.

Syntax: PENUP [*Kanal*]
PENDOWN [*Kanal*]

Beispiel:
a) `PENUP` {Hebt den Stift in dem Standard-Kanal ab}
b) `PENDOWN#2` {Setzt den Stift in dem Fenster auf, das mit Kanal 2 verbunden ist}

PI
Mathematische Funktionen

PI ist eine Funktion, die den Wert von π zurückgibt.

Syntax: PI

Beispiel: `PRINT PI`

POINT
POINT_R
Grafik

Mit **POINT** wird ein Punkt bei der angegebenen Position in dem *Fenster* gezeichnet, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist. Der Punkt wird bezogen auf den *Ursprung des grafischen Koordinatensystems* gezeichnet. Mit **POINT_R** werden sämtliche Punkte bezogen auf den Grafik-Cursor angegeben, der durch jeden gezeichneten Punkt neu gesetzt wird.

Mit einem einzigen Aufruf von **POINT** können mehrere Punkte gezeichnet werden.

Syntax: *x:= numerischer_Ausdruck*
y:= numerischer_Ausdruck

Parameter:= x, y

POINT [*Kanal,*] *Parameter* * [, *Parameter*]*

Beispiel:

a) `POINT 256,128` {Zeichnet einen Punkt bei (256,128)}

b) `POINT x, x*x` {Zeichnet einen Punkt bei (x,x*x)}

c)
```
100 REPeat beispiel
110   INK RND(255)
120   POINT RND(100),RND(100)
130 END REPeat beispiel
```

POKE
POKE_W
POKE_L
BASIC

Mit **POKE** kann der Inhalt von Speicherplätzen geändert werden. Bei Zugriffen auf Worte und lange Worte muß eine geradzahlige Adresse angegeben werden.

POKE verfügt über drei Formen, mit denen auf ein Byte (8 Bit), ein Wort (16 Bit) bzw. ein langes Wort (32 Bit) zugegriffen wird.

Syntax: *Adresse:= numerischer_Ausdruck*
Daten:= numerischer_Ausdruck

POKE *Adresse, Daten* {Zugriff auf ein Byte}
POKE_W *Adresse, Daten* {Zugriff auf ein Wort}
POKE_L *Adresse, Daten* {Zugriff auf ein langes Wort}

Beispiel:

a) `POKE 12235,0` {Setzt das Byte bei 12235 auf 0}

b) `POKE_L 131072, 12345` {Setzt das lange Wort bei 131072 auf 12345}

Hinweis

Werden Daten in Speicherbereichen geändert, die von Qdos benutzt werden, so kann dies zu Systemabsturz und Datenverlust führen. Von der Änderung derartiger Bereiche wird abgeraten.

PRINT
Einheiten
Microdrives

Mit diesem Befehl kann die Ausgabe an den angegebenen *Kanal* oder Standard-*Kanal* gesendet werden. Normalerweise werden mit **PRINT** Daten auf dem *Bildschirm* des QL angezeigt.

Syntax: *Trennzeichen:=* | !
| ,
| \
| ;
| TO numerischer_Ausdruck

Posten:= | *Ausdruck*
| *Kanal*
| *Trennzeichen*

PRINT * [*Posten*]*

Mehrere *Trennzeichen* sind beim Drucken zulässig. Zwischen der *Kanal-Angabe* und den *Ausdrücken* muß mindestens ein Trennzeichen stehen.

Beispiel:

a) `PRINT ''Guten Tag Allerseits''`
{Druckt "Guten Tag Allerseits" in der Standard-Ausgabeeinheit (Kanal 1) aus.}

b) `PRINT #5, ''Daten'', 1,2,3,4`
{Gibt die angegebenen Daten an Kanal 5 aus (der vorher geöffnet worden sein muß)}

c) `PRINT TO 20; ''Dies ist Spalte 20''`

Trennzeichen

! Normalerweise wird zwischen den auf dem Bildschirm ausgegebenen Elementen ein Leerzeichen eingefügt. Paßt ein Element nicht in die laufende Zeile, so wird eine neue Zeile begonnen. Befindet sich die laufende Druckposition am Anfang einer Zeile, so wird kein Leerzeichen ausgegeben. **!** bezieht sich auf das nächste auszudruckende Element und muß demzufolge davor gesetzt werden. Außerdem muß ein **;** oder ein **!** an das Ende der Druckliste gesetzt werden, wenn diese Art der Ausgabesteuerung über eine Reihe von **PRINT**-Anweisungen fortgesetzt werden sollen.

, Normales Trennzeichen, SuperBASIC setzt in der Ausgabe einen Tabulator in Abständen von acht Spalten.

\ Beginnt eine neue Zeile.

; Beläßt die Druckposition unmittelbar hinter dem letzten auszudruckenden Element. Die Ausgabe der nächsten **PRINT**-Anweisung beginnt in derselben Zeile.

TO Mit diesem Trennzeichen wird eine Tabulator-Operation ausgeführt. Folgt auf **TO** ein *numerischer_Ausdruck*, so wird die Druckposition in die Spalte gesetzt, die mit *numerischer_Ausdruck* angegeben wird. Gibt es die angeforderte Spalte nicht oder liegt sie vor der laufenden Druckposition so ist das Trennzeichen wirkungslos.

RAD
Mathematische
Funktionen

RAD ist eine Funktion, mit der ein in Grad angegebener Winkel in das Bogenmaß (Radiant) umgerechnet wird.

Syntax: RAD (*numerischer_Ausdruck*)

Beispiel: `PRINT RAD(180)` {Druckt 3.141593 aus}

RANDOMISE

Mathematische Funktionen

Mit **RANDOMISE** kann ein Startwert des Zufallszahlengenerators festgelegt werden. Wird ein Parameter angegeben, so dient er als neuer Startwert. Wird kein Parameter angegeben, so wird der neue Startwert anhand interner Daten festgelegt.

Syntax: RANDOMISE [*numerischer_Ausdruck*]

Beispiel:

a) `RANDOMISE` {Der Startwert wird anhand interner Daten festgelegt}

b) `RANDOMISE 3.2235` {Der Startwert wird auf 3.2235 festgelegt}

RECOL

Farbe, Fenster

Mit **RECOL** wird die *Farbe* der einzelnen Pixel in dem *Fenster*, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist, entsprechend einer beliebigen Vorgabe neu festgelegt. Der erste Parameter gibt die neue Farbe für alle schwarzen Pixel an, der zweite Parameter die neue Farbe für die blauen Pixel usw.

Bei der Farbangabe muß es sich um eine Vollfarbe handeln, d.h. sie muß in dem Bereich von 0 bis 7 liegen.

Syntax:

c0: = neue Farbe für Schwarz
c1: = neue Farbe für Blau
c2: = neue Farbe für Rot
c3: = neue Farbe für Magenta
c4: = neue Farbe für Grün
c5: = neue Farbe für Cyan
c6: = neue Farbe für Gelb
c7: = neue Farbe für Weiß

RECOL [*Kanal,*] *c0, c1, c2, c3, c4, c5, c6, c7*

Beispiel: `RECOL 2,3,4,5,6,7,1,0` {Ändert Schwarz in Rot, Blau in Magenta, Rot in Grün, Magenta in Cyan usw.}

REMark

Mit **REMark** kann erläuternder Text in ein Programm eingefügt werden. Der Rest der Zeile wird von SuperBASIC ignoriert.

Syntax: **REMark** *Text*

Beispiel: `REMark Dies ist ein Kommentar in einem Programm`

Kommentar

Mit **REMark** werden Kommentare zu einem Programm hinzugefügt, um dieses besser lesbar zu gestalten.

RENUM

Mit **RENUM** können eine oder mehrere Gruppen von SuperBASIC-Zeilennummern geändert werden. Werden keine Parameter angegeben, so numeriert **RENUM** das ganze Programm neu. Die neue Auflistung beginnt bei Zeile 100 und numeriert in Zehnerschritten neu.

Wird eine Anfangszeile angegeben, so bleiben die *Zeilennummern* vor der Anfangszeile unverändert. Wird eine Endzeile angegeben, so bleiben die Zeilennummern hinter dieser Endzeile unverändert.

Werden eine Anfangsnummer und eine Schrittweite angegeben, so werden die neu zu numerierenden Zeilen ab der Anfangsnummer numeriert, wobei in Schritten der angegebenen Weite neu numeriert wird.

Enthält eine **GOTO**- oder **GOSUB**-Anweisung einen Ausdruck, der mit einer Zahl beginnt, so wird diese Zahl als Zeilennummer behandelt und neu numeriert.

Syntax:

Startzeile:= *numerischer_Ausdruck* {Beginn der Neunumerierung}
Endzeile:= *numerischer_Ausdruck* {Ende der Neunumerierung}
Anfangsnummer:= *numerischer_Ausdruck* {Erste neue Zeilennummer}
Schritt:= *numerischer_Ausdruck* {Schrittweite}

RENUM [*Startzeile* [**TO** *Endzeile*];] [*Startnummer*] [,*Schrittweite*]

Beispiel:

a) `RENUM` {Numeriert das ganze Programm in Zehnerschritten ab 100 neu}

b) `RENUM 100 TO 200` {Numeriert in Zehnerschritten von 100 bis 200 neu}

Hinweis

Es darf nicht versucht werden, Programmzeilen mit **RENUM** außerhalb der vorgegebenen Reihenfolge neu zu numerieren, d. h. Zeilen an eine andere Stelle in dem Programm zu setzen. **RENUM** darf in einem Programm nicht benutzt werden.

REPeat
END REPeat

Wiederholung

Mit **REPeat** können allgemeine *Schleifen* erstellt werden. Um eine maximale Wirkung zu erzielen, muß **REPeat** mit **EXIT** benutzt werden. **REPeat** kann sowohl in einer langen als auch in einer kurzen Form benutzt werden.

Kurze Form

Auf das Schlüsselwort **REPEAT** und den Schleifen-Namen folgen auf derselben logischen Zeile ein Doppelpunkt und eine Reihe von SuperBASIC-Anweisungen. Durch **EXIT** wird die normale Verarbeitung bei der nächsten logischen Zeile wieder aufgenommen.

Syntax: **REPeat** *Name:Anweisungen*

Beispiel:

```
REPeat warte: IF INKEY$ <> '' '' THEN EXIT warte
```

Lange Form

Das Schlüsselwort **REPEAT** und der Schleifen-Name sind die einzigen Anweisungen in der logischen Zeile. Nachfolgende Zeilen enthalten eine Reihe von SuperBASIC-*Anweisungen,* die mit einer **END REPeat**-Anweisung beendet werden.

Die Anweisungen zwischen **REPeat** und **END REPeat** werden wiederholt von SuperBASIC verarbeitet.

Syntax:

REPeat *Name*
  Anweisungen
END REPeat *Name*

Beispiel:

```
100 LET nummer = RND(1 TO 50)
110 REPeat tip
120   INPUT ''Geben Sie Ihren Tip'', tip
130   IF tip = nummer THEN
140     PRINT ''Sie haben richtig geraten''
150     EXIT tip
160   ELSE
170     PRINT ''Sie haben falsch geraten''
180  END IF
190 END REPeat tip
```

Kommentar

Normalerweise ist mindestens eine Anweisung in einer **REPeat**-Schleife eine **EXIT**-Anweisung.

RESPR

Qdos

RESPR ist eine Funktion, mit der Platz für residente Prozeduren im Speicher zugeordnet wird (beispielsweise um die SuperBASIC-Prozedurliste zu erweitern).

Syntax: *Platz:= numerischer_Ausdruck*

RESPR (*Platz*)

Beispiel:

```
PRINT RESPR(1024)
```

{Druckt die Basisadresse eines 1.024 Bytes umfassenden Blockes aus}

RETurn

Funktionen und Prozeduren

Mit **RETurn** wird eine *Funktion* oder *Prozedur* beendet und die Verarbeitung bei der Anweisung fortgesetzt, die auf den Prozedur- oder Funktionsaufruf folgt. Wird **RETurn** innerhalb einer Funktionsdefinition benutzt, so wird mit dieser Anweisung der Wert der Funktion zurückgegeben.

Syntax: RETurn [*Ausdruck*]

Beispiel:

a)
```
100 PRINT bink (3,3)
110 DEFine FuNction bink (m,n)
120   IF m = 0 THEN RETurn n + 1
130   IF n = 0 THEN RETurn bink (m-1,1)
140   RETurn bink(m-1,bink(m,n-1))
150 END DEFine
```

b)
```
100 LET warn_signal = 1
110 LET fehler_nummer = RND(0 TO 10)
120 warnung fehler_nummer
130 DEFine PROCedure warnung(n)
140   IF warn_signal THEN
150     PRINT ''Warnung:''
160     SELect ON n
170       ON n = 1
180         PRINT ''Microdrive voll''
190       ON n = 2
200         PRINT ''Datenbereich voll''
210       ON n = REMAINDER
220         PRINT ''Programmfehler''
230     END SELect
240   ELSE
250     RETurn
260   END IF
270 END DEFine
```

Kommentar

RETurn muß in einer Prozedur nicht unbedingt vorhanden sein. Erreicht die Verarbeitung die **END DEFine**-Anweisung eine Prozedur, so wird die Prozedur automatisch beendet.

RETurn alleine wird für den Rücksprung aus einer **GOSUB**-Anweisung benutzt.

RND

Mathematische Funktionen

Mit **RND** wird eine Pseudo-Zufallszahl erzeugt. Bis zu zwei Parameter können für **RND** angegeben werden. Werden keine Parameter angegeben, so gibt **RND** eine Pseudo-Zufallszahl in Form einer *Gleitkommazahl* im Bereich von 0 bis 1 zurück. Wird ein einziger Parameter angegeben, so gibt **RND** eine ganze Zahl in dem Bereich von 0 bis zum angegebenen Parameter einschließlich zurück. Werden zwei Parameter angegeben, so gibt **RND**eine ganze Zahl in dem von den beiden Parametern angegebenen Bereich zurück.

Syntax: RND ([*numerischer_Ausdruck*] [TO *numerischer_Ausdruck*])

Beispiel:

a)	`PRINT RND`	{Gleitkommazahl zwischen 0 und 1}
b)	`PRINT RND(10 TO 20)`	{Ganze Zahl zwischen 10 und 20}
c)	`PRINT RND(1 TO 6)`	{Ganze Zahl zwischen 1 und 6}
c)	`PRINT RND(10)`	{Ganze Zahl zwischen 0 und 10}

RUN

Programm

Mit **RUN** kann ein SuperBASIC-Programm gestartet werden. Wird in dem **RUN**-Befehl eine *Zeilennummer* angegeben, so wird das Programm an dieser Stelle gestartet. Ansonsten startet das Programm bei der niedrigsten Zeilennummer.

Syntax: RUN [*numerischer_Ausdruck*]

Beispiel:

a) `RUN` {Ausführung ab Programmanfang}
b) `RUN 10` {Ausführung ab Zeile 10}
c) `RUN 2*20` {Ausführung ab Zeile 40}

Kommentar

Obwohl **RUN** innerhalb eines Programms benutzt werden kann, wird dieser Befehl normalerweise für den Start der Programmausführung benutzt, indem er als *direkter Befehl* eingegeben wird.

SAVE

Einheiten Microdrives

Mit **SAVE** wird ein SuperBASIC-Programm auf einer QL-Einheit gesichert.

Syntax:

Zeile: = | *numerischer_Ausdruck* TO *numerischer_Ausdruck* 1
| *numerischer_Ausdruck* TO 2
| TO *numerischer_Ausdruck* 3
| *numerischer_Ausdruck* 4
| 5

SAVE *Einheit* *[,Zeile]*

1 – sichert von der angegebenen Zeile bis zur angegebenen Zeile.
2 – sichert von der angegebenen Zeile bis zum Ende
3 – sichert vom Anfang bis zu der angegebenen Zeile
4 – sichert die angegebene Zeile
5 – sichert das ganze Programm.

Beispiel:

a) `SAVE mdv1_programm, 20 TO 70`
{Sichert die Zeilen 20 bis 70 in mdv1_programm}

b) `SAVE mdv2_test_programm, 10,20,40`
{Sichert die Zeilen 10, 20, 40, in mdv1_test_programm}

c) `SAVE neto_3`
{Sichert das ganze Programm in dem Netzwerk}

d) `SAVE ser1`
{Sichert das ganze Programm in der seriellen Einheit 1}

SBYTES
Einheiten
Microdrives

Mit **SBYTES** können Bereiche des QL-Speichers in einer QL-*Einheit* gesichert werden.

Syntax:

Startadresse:= numerischer_Ausdruck
Länge:= numerischer_Ausdruck

SBYTES *Einheit, Startadresse, Länge*

Beispiel:

a) `SBYTES mdv1_bildschirm_daten, 131072,32768`
{Sichert ab Adresse 131072, Länge 32768 Bytes, in mdv1_bildschirm_daten}

b) `SBYTES mdv1_test_programm, 50000,10000`
{Sichert ab Adresse 50000, Länge 10000 Bytes, in mdv_test_programm}

c) `SBYTES neto_3, 32768, 32768`
{Sichert ab Adresse 32768, Länge 32768 Bytes, in dem Netzwerk}

d) `SBYTES ser1, 0,32768`
{Sichert ab Adresse 0, Länge 32768 Bytes, in dem seriellen Kanal 1}

SCALE
Grafik
Koordinaten

Mit **SCALE** können Maßstabsfaktoren und Lage des Ursprungs im *grafischen Koordinatensystem* geändert werden. Ein Maßstabsfaktor z bedeutet, daß die y-Achse im betreffenden *Fenster* in z Einheiten unterteilt wird. Die Anzahl der Koordinateneinheiten der x-Achse richtet sich nach der Form des Fensters. Der Standardwert für den Maßstabsfaktor ist 100. Der Ursprung wird standardmäßig in der linken unteren Ecke des Fensters festgelegt. Durch Verlegung des Ursprungs können Sie das Fenster nach Belieben auf den gewünschten Ausschnitt der Zeichenfläche legen.

Syntax:

x:= numerischer_Ausdruck
y:= numerischer_Ausdruck

Ursprung:= x, y
Maßstab:= numerischer_Ausdruck

SCALE[*Kanal,*] *Maßstab, Ursprung*

Beispiel:

a) `SCALE .5 ,.1,.1`
{Legt den Maßstab auf 0,5 mit dem Ursprung bei 0.1,0.1 fest}

b) `SCALE 10,0,0`
{Legt den Maßstab auf 10 mit dem Ursprung bei 0,0 fest}

c) `SCALE 100,50,50`
{Legt den Maßstab auf 100 mit dem Ursprung bei 50,50 fest}

SCROLL
Fenster

Mit **SCROLL** wird das *Fenster,* das mit dem angegebenen *Kanal* oder Standard-*Kanal* verbunden ist, um die angegebene Anzahl von Pixeln nach oben oder unten gerollt. *Paper* wird oben oder unten auf den Bildschirm gerollt, um den gelöschten Platz auszufüllen.

Soll nur ein Teil des Bildschirminhalts gerollt werden, so kann ein dritter Parameter angegeben werden.

Syntax: *Teil:=* *numerischer_Ausdruck*
Entfernung:= *numerischer_Ausdruck*

Teil = 0 – Ganzer Bildschirm (standardmäßig braucht dann kein Parameter angegeben zu werden)
Teil = 1 – Bildschirmanfang, ohne Cursor-Zeile
Teil = 2 – Bildschirmende, ohne Cursor-Zeile

SCROLL [*Kanal,*] *Entfernung* [, *Teil*]

Ist die Entfernung positiv, so wird der Bildschirminhalt nach unten geschoben.

Beispiel:

a) `SCROLL 10` {Der Bildschirminhalt wird um 10 Pixel nach unten gerollt}
b) `SCROLL -70` {Der Bildschirminhalt wird um 70 Pixel nach oben gerollt}
c) `SCROLL -10,2` {Der untere Teil des Fensters wird um 10 Pixel nach oben gerollt}

SDATE
Uhr

Mit dem **SDATE**-Befehl kann die *Uhr* des QL eingestellt werden.

Syntax: *Jahr:=* *numerischer_Ausdruck*
Monat:= *numerischer_Ausdruck*
Tag:= *numerischer_Ausdruck*
Stunden:= *numerischer_Ausdruck*
Minuten:= *numerischer_Ausdruck*
Sekunden:= *numerischer_Ausdruck*

SDATE *Jahr, Monat, Tag, Stunden, Minuten, Sekunden*

Beispiel:

a) `SDATE 1984,3,21,0,0,0`
b) `SDATE 1984,1,12,9,30,0`
c) `SDATE 1984,4,2,0,0,0`

SELect
END SELect
Bedingungen

Mit **SELect** können je nach Wert der Auswahlvariablen verschiedene Programmteile verarbeitet werden.

Definition: *Auswahl_Variable:=* *numerische_Variable*

Auswahl_Elemente:= |*Ausdruck*
|*Ausdruck* **TO** *Ausdruck*

Auswahl_Liste:= |*Auswahl_Element**[,*Auswahl_Element*]*

Lange Form

Mit diesem Befehl können mehrere Aktionen je nach Wert einer *Auswahl_Variablen* ausgewählt werden. Bei der Auswahlvariablen handelt es sich um das letzte Element auf der logischen Zeile. Darauf folgt eine Reihe von SuperBASIC-*Anweisungen,* die mit der nächsten **ON**-Anweisung oder der **END SELect**-Anweisung beendet wird. Handelt es sich bei dem Auswahlelement um einen Ausdruck, so wird innerhalb etwa eines Teils von 10^{-7} eine Prüfung vorgenommen. Ansonsten wird für "Ausdruck **TO** Ausdruck" der Bereich genau und vollständig getestet. Die **ON REMAINDER**-Anweisung wird benutzt, wenn keine der anderen Auswahlbedingungen erfüllt wird.

Syntax:

SELect ON *Auswahl_Variable*
*[[**ON** *Auswahl_Variable*] = *Auswahl_Liste*
Anweisungen]*
[**ON** *Auswahl_Variable* = **REMAINDER**
Anweisungen
END SELect

Beispiel:

```
100 LET fehler_zahl = RND(1 TO 10)
110 SELect ON fehler_zahl
120   ON fehler_zahl = 1
130     PRINT ''Division durch Null''
140     LET fehler_zahl = 0
150   ON fehler_zahl = 2
160     PRINT ''Datei nicht gefunden''
170     LET fehler_zahl = 0
180   ON fehler_zahl = 3 TO 5
190     PRINT ''Microdrive-Datei nicht gefunden''
200     LET fehler_zahl = 0
210   ON fehler_zahl = REMAINDER
220     PRINT ''Unbekannter Fehler''
230 END SELect
```

Wird die Auswahlvariable im Hauptteil der **SELect**-Anweisung benutzt, so muß sie mit der Auswahlvariablen übereinstimmen, die in dem Auswahlkennsatz angegeben wird.

Kurze Form

Mit der kurzen Form der **SELect**-Anweisung kann eine einzelne Zeile ausgewählt werden. In derselben logischen Zeile, in der auch die **SELect**-Anweisung steht, folgt eine Reihe von SuperBASIC-Anweisungen. Wird die Bedingung in der Auswahlanweisung erfüllt, so wird die Folge mit SuperBASIC-Anweisungen verarbeitet.

Syntax:

SELect ON
Auswahl_Variable = *Auswahl_Liste: Anweisung* *[: *Anweisung*]*

Beispiel:

a)
```
SELect ON test_daten = 1 TO 10:
PRINT ''Antwort innerhalb des Bereichs''
```
b)
```
SELect ON antwort = 0.00001 TO 0.00005
PRINT ''Genauigkeit in Ordnung''
```
c)
```
SELect ON a = 1 TO 10: PRINT a ! ''im Bereich''
```

Kommentar

Mit der Kurzform der **SELect**-Anweisung können Bereiche einfacher getestet werden, als mit einer **IF**-Anweisung. Hierzu sollte Beispiel b oben mit der entsprechenden **IF**-Anweisung verglichen werden.

SEXEC
Qdos

Mit diesem Befehl wird ein Speicherbereich in einer Form gesichert, in der er mit dem **EXEC**-Befehl geladen und ausgeführt werden kann.

Die gesicherten Daten müssen ein Programm im Maschinencode darstellen.

Syntax: *Startadresse:=* *numerischer_Ausdruck* {Beginn des Bereichs}
Länge:= *numerischer_Ausdruck* {Länge des Bereichs}
Datenbereich:= *numerischer_Ausdruck* {Länge des Datenbereichs, der von dem Programm benötigt wird}

SEXEC *Einheit, Startadresse, Länge, Datenbereich*

Beispiel: `SEXEC mdv1_programm, 262144,3000,500`

Kommentar

Bevor mit diesem Befehl gearbeitet wird, sollte unbedingt die Qdos-Systemdokumentation gelesen werden.

SIN
Mathematische Funktionen

Mit **SIN** wird der Sinus des angegebenen Parameters berechnet. Der Parameter wird als Winkel im Bogenmaß aufgefaßt.

Syntax: *Winkel:=* *numerischer_Ausdruck* {Bereich –10000 .. 10000 in Radiant}

SIN (*Winkel*)

Beispiel:
a) `PRINT SIN(3)`
b) `PRINT SIN(3.141592654/2)`

SQRT
Mathematische Funktionen

Mit diesem Befehl wird die Quadratwurzel des angegebenen Argumentes berechnet. Das Argument muß größer oder gleich Null sein.

Syntax: SQRT(*numerischer_Ausdruck*) {Bereich > = 0}

Beispiel:

a) `PRINT SQRT(3)` {Druckt die Quadratwurzel von 3 aus}

b) `LET C = SQRT(a^2+b^2` {Hier wird c gleich der Quadratwurzel von $a^2 + b^2$}

STOP
BASIC

Mit **STOP** wird die Ausführung eines Programms beendet. SuperBASIC kehrt wieder zu dem *Befehls-Interpreter* zurück.

Syntax: STOP

Beispiel:

a) `STOP`

b) `IF n = 100 THEN STOP`

Nach **STOP** kann ein **CONTINUE**-Befehl ausgeführt werden.

Kommentar

Die letzte ausführbare Zeile eines Programms entspricht einem automatischen Stopp.

STRIP

Fenster

Mit **STRIP** wird die aktuelle *Streifenfarbe* in dem Fenster festgelegt, das mit dem angegebenen *Kanal* oder Standard-*Kanal* verbunden ist. Die *Streifenfarbe* entspricht der Hintergrundfarbe, die benutzt wird, wenn **OVER 1** ausgewählt wird. Wird **PAPER** gesetzt, so wird die *Streifenfarbe* automatisch auf die neue **PAPER**-Farbe gesetzt.

Syntax: **STRIP** [*Kanal,*] *Farbe*

Beispiel:

a) `STRIP 7` {Legt einen weißen Streifen fest}
b) `STRIP 0,4,2` {Legt einen Streifen mit einem schwarz/grünen Punktmuster fest}

Kommentar

Die Benutzung von **STRIP** kann mit der Benutzung eines Highlighting verglichen werden.

TAN

Mathematische Funktionen

TAN berechnet den Tangens des angegebenen Argumentes. Das Argument muß in dem Bereich von –30000 bis 30000 liegen und muß in Radiant angegeben werden.

Syntax: **TAN** (*numerischer_Ausdruck*) {Bereich –30000 .. 30000}

Beispiel:

b) `TAN(3)` {Druckt tan 3 aus}
c) `TAN(3.141592654/2)` {Druckt tan $\pi/2$ aus}

TRA
QDOS

Mit **TRA** können Sie festlegen, welche Art von *Zeichensatz* das mit einer der beiden seriellen Anschlüsse *ser1* oder *ser2* verbundene Gerät erwartet. **TRA 1** wird gewählt, wenn das Gerät auf den *deutschen Zeichensatz* eingestellt ist, **TRA 0** sonst.

Syntax: *x:= ganzzahliger_Ausdruck* (Bereich 0 oder 1)

TRA x

Beispiel: `TRA 1` stellt den deutschen Zeichensatz ein

Hinweis

Der QL-Printer druckt den deutschen Zeichensatz. Sie brauchen daran aber keine gesonderten Einstellungen vorzunehmen. Deshalb müssen Sie ihn mit **TRA 0** ansteuern.

TURN
TURNTO
Turtle-Grafik

Mit **TURN** wird die Richtung der 'Turtle' um einen angegebenen Winkel gedreht, während die Turtle mit **TURNTO** in eine bestimmte Richtung gedreht werden kann.

Die Turtle wird in dem *Fenster* gedreht, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

Der Winkel wird in Grad angegeben. Wird eine positive Gradzahl angegeben, so wird die Turtle entgegen dem Uhrzeigersinn gedreht. Bei einer negativen Zahl wird sie im Uhrzeigersinn gedreht.

Ursprünglich zeigt die Turtle auf 0°, d.h. auf die rechte Seite des Fensters.

Syntax: *Winkel:= numerischer_Ausdruck* {Winkel in Grad}

TURN [*Kanal,*] *Winkel*
TURNTO [*Kanal,*] *Winkel*

Beispiel:
a) `TURN 90` {Dreht um 90°}
b) `TURNTO 0` {Dreht in Richtung 0°}

UNDER
Fenster

Mit diesem Befehl wird die *Unterstreichung* für die nachfolgenden Ausgabezeilen ein- oder ausgeschaltet. Die *Unterstreichung* wird mit der aktuellen **INK**-Farbe des *Fensters* vorgenommen, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist.

Syntax: *Schalter:= numerischer_Ausdruck* {Bereich 0..1}

UNDER [*Kanal,*] *Schalter*

Beispiel:

b) `UNDER 1` {Schaltet die Unterstreichung ein}
c) `UNDER 0` {Schaltet die Unterstreichung aus}

WIDTH
Einheiten

Mit **WIDTH** kann die Standardbreite für Nicht-Konsoleinheiten angegeben werden, z. B. für Drucker.

Syntax: *Zeilen_Breite:= numerischer_Ausdruck*

WIDTH [*Kanal,*] *Zeilen_breite*

Beispiel:

a) `WIDTH 80` {Legt die Zeilenbreite bei der Einheit auf 80 fest}
b) `WIDTH #6,72` {Legt die Zeilenbreite bei der mit Kanal 6 verbundenen Einheit auf 72 fest}

WINDOW
Fenster

Mit diesem Befehl kann der Benutzer die Position und Größe des *Fensters* ändern, das mit dem angegebenen *Kanal* oder dem Standard-*Kanal* verbunden ist. Eventuelle Umrahmungen werden gelöscht, wenn das *Fenster* neu definiert wird.

Die Koordinaten werden mit dem *Pixel-System,* bezogen auf die linke obere Ecke des Maximalfensters, angegeben.

Syntax:

Breite:= numerischer_Ausdruck
Höhe:= numerischer_Ausdruck
x:= numerischer_Ausdruck
y:= numerischer_Ausdruck

WINDOW [*Kanal,*] *Breite, Höhe, x, y*

Beispiel:

a) `WINDOW 30, 40, 10, 10` {Fenster mit 30 x 40 Pixeln bei 10,10}

A
ABS ... 1
Absolute Werte ... 1
ACOT ... 1
ADATE ... 2
Uhr ... 2,52
ARC ... 2
Kontangens ... 2
Tangens ... 2
ARC_R ... 2
Arctangens ... 2
Argumente ... 11,54
DIM ... 18
DIMN ... 18
Tabellen ... 18
AT ... 3
ATAN ... 1
Auflösung ... 36
Ausdruck ... 15
Ausfüllen ... 23
AUTO ... 3
Anweisung ... 37

B
BAUD ... 4
Baudrate ... 4
BEEP ... 4
BEEPING ... 4
Betriebsart, niedrigauflösend ... 24
Befehl, direkter ... 50
Befehls-Interpreter ... 55
Bildschirm ...
BLOCK ... 5
BORDER ... 5
FLASH ... 24
INK ... 27
MODE ... 36
OVER ... 40
PAN ... 41
PAPER ... 41
PRINT ... 45
RECOL ... 52
STRIP ... 56
UNDER ... 58
WINDOW ... 59
BLOCK ... 5
BIRDER ... 6

C
CALL ... 6
CHR$ ... 7
CODE ... 9
CIRCLE ... 7
CIRCLE_R ... 7
grafisches Koordinatensystem ... 7,32,44,51
CLEAR ... 8
BASIC ... 8
CLOSE ... 8
Fenster ... 9
Kanal ... 8
CLS ... 9
Fenster ... 9
Kanal ... 9
CODE ... 9
CHR$ ... 9,7
CONTINUE ... 10
Programm ... 10
RETRY ... 10
COPY ... 10
COPY_N ... 10
COS ... 11
Kosinus ... 11
COT ... 11
Kotangens ... 11
CSIZE ... 12
Fenster ... 12
Kanal ... 12
Schriftgröße ... 12
CURSOR ... 12
Fenster ... 12
Kanal ... 12
Pixel-Koordintensystem ... 3,12

D
DATA ... 13
READ ... 13
RESTORE ... 13
DATE ... 14
DATE$ ... 14
Uhr ... 14
Datei
COPY ... 10
COPY_N ... 10
DELETE ... 17
DIR ... 19
LBYTES ... 31
LOAD ... 33
LRUN ... 35
MERGE ... 35
MRUN ... 37
OPEN ... 40
OPEN_IN ... 40
OPEN_NEW ... 40
PRINT ... 45
RUN ... 50
SAVE ... 50
DAY$ ... 14
DEFine ... 15
FuNction ... 15
aktuelle Paramter ... 15,16
formale Parameter ... 15,16
PROCedure ... 16
LOCal ... 16
RETurn ... 15
DEG ... 17
DELETE ... 17
Einheiten ... 17
deutscher Zeichensatz ... 57
DIM ... 18
Ganze Zahlen ... 18
Gleitkomma Zahlen ... 18
Index ... 18
Strings ... 18
DIMN ... 18
DIR ... 19
Einheiten ... 19
Inhaltsverzeichnis ... 19
direkter Befehl ... 37,50

DIV 19
DLINE 20

E

E/A ....
INKEY$ 28
KEYROW 30
PRINT 45
EDIT 20
Einheiten
CLOSE 8
FORMAT 25
LBYTES 31
LOAD 33
LRUN 35
MERGE 35
MRUN 37
NET 38
OPEN 40
RUN 50
SAVE 50
SBYTES 41
ELLIPSE 7
ELLIPSE_R 7
END 15,16
DEFine 15,16
FOR 24
NEXT 24
Schleifennachsatz 24
STEP 24
TO 24
EOF 21
Erhöhung 20
EXEC 21
EXEC_W 21
EXIT 22
FOR 22
REPeat 22

F

Farbe
INK 27
MODE 36
PAPER 36
RECOL 46
Fenster ....
CLS 9
DIR 19
INK 27
MODE 36
RECOL 46
SCALE 51
SCROLL 52
TURN 58
TURNTO 58
WINDOW 59
FILL 23
Ausfüllen 23
Grafik 23
Turtle-Grafik 23
FILL$ 23
FLASH 24
Blinken 24
niedrigauflösende Betriebsart 24
FN 15
FOR 24
mit EXIT 24
mit NEXT 24
Schleifennachsatz 24
FORMAT 25
FuNction 15
DEFine 15
RETurn 49

G

ganze Zahlen 18
Gleitkomma-Zahlen 18
GOSUB 25
GOTO 26
Grafik
ARC 2
ARC_R 2
CIRCLE 7
CIRCLE_R 7
ELLIPSE 7
ELLIPSE_R 7
FILL 23
LINE 32
LINE_R 32
POINT 44
POINT_R 44
SCALE 51
Grafik Cursor 32
Grafik Koordinaten 32
grafisches Koordinatensystem
CIRCLE 7
ELLIPSE 7
LINE 32
POINT 44
SCALE 51

H

Highlighting 56
Hintergrundfarbe 41

I

IF 26,27
INK 27
INEY$ 28
INPUT 28
INSTR 29
Index 18
INT 29

K

Kanal
CLOSE 8
FORMAT 25
LBYTES 31
LIST 33
LOAD 33
LRUN 35
MERGE 35
MRUN 37

NET 38
OPEN 40
OPEN_IN 40
OPEN_NEW 40
RUN 50
SAVE 50
SBYTES 51
KEYROW 30

L

LBYTES 31
LEN 31
Stringausdruck 31
LET 32
LINE 32
Grafik Cursor 32
Grafik Koordinatensystem 32
LINE_R 32
LIST 33
LN 34
LOAD 33
MISTake 33
Programm 33
LOG10 34
LOCal 34
Funktion 34
Prozedur 34
Tabellen 34
Variable 34
LRUN 35

M

Maschinencode 6
CALL 6
EXEC 21
EXEC_W 21
SEXEC 54
Mathematische Funktionen
ABS 1
absolute Weite 1
ACOS 1
Arcuskosinus 1
ACOT 1
Arcuskotangens 1
ASIN 1
Arcussinus 1
ATAN 1
Arcustangens 1
COS 11
Kosinus 11
COT 11
Kotangens 11
EXP 22
Exponent 22
INT 29
Exponent 22
INT 29
LOG10 34
LN 34
RAD 45
SIN 54
Sinus 54
SQR 55
Quadratwurzel 55
TAN 56
Tangens 56
Microdrives
COPY 10
DELETE 17
FORMAT 25
LOAD 33
MISTake 35
LOAD 33
MERGE 35
LRUN 35
MODE 36
MOVE 37
MRUN 37
Multitasking
PAUSE 42
SEXEC 54

N

NET 38
Network 38
NEW 38
NEXT 39
Schleifennachsatz 39
mit FOR 24
mit REPeat 48
niedrigauflösende Betriebsart 24

O

ON GOSUB 39
Programmzeile 39
ON GOTO 39
Programmzeile 39
OPEN 40
Kanal 40
serielles Port 40
Fenster 40
OPEN_IN 40
OPEN_NEW 40
Operatoren
INSTR 29
MOD 36
OVER 40
Überschreiben 40

P

PAN 40
Bildschirm 40
PAPER 41
Hintergrundfarbe 41
Parameter 15,16
PAUSE 42
PEEK 42
PEEK_L 42
PEEK_W 42
PENDOWN 43
Turtle-Grafik 43
PENUP 43
Turtle-Grafik 43
PI 43

Pixel-Koordinatensystem
- AT ... 3
- CURSOR ... 12

PRINT ... 45
- OVER ... 40
- UNDER ... 58

Programm
- CONTINUE ... 10
- LOAD ... 33
- LRUN ... 35
- MRUN ... 37
- PAUSE ... 42
- PETRY ... 10
- RUN ... 50
- SAVE ... 50

Programmzeile
- GOSUB ... 39
- GOTO ... 39

Prozeduren
- DEFine ... 15,16
- LOCal ... 34
- RETurn ... 49

R

RAD ... 45
RANDOMISE ... 46
- RND ... 49

READ ... 12
RECOL ... 47
REM ... 47
REMark ... 47
RENUM ... 47
- Zeilennummer ... 47

REPeat ... 48
- Schleifen ... 48
- EXIT ... 22
- NEXT ... 39

Reset ... 8
Reset Uhr ... 2,52
RESPR ... 48
RESTORE ... 13
RETRY ... 10
RETurn ... 49
- mit FuNction ... 15
- mit PROCedur ... 16

RS-232-C ... 4
BAUD ... 4
RND ... 49
RUN ... 50
- LRUN ... 35
- Zeilennummer ... 50

S

SAVE ... 50
SBYTES ... 51
SCALE ... 51
Schleifen ... 48
Schleifennachsatz ... 24,29
SCROLL ... 52
SDATE ... 52
- ADATE ... 2

SELect ... 53
SINE ... 54
- Sinus ... 54

SQRT ... 55
- Quadratwurzel ... 55

Start Programm ... 35,50
Stations Nummer ... 38
STOP ... 55
String
- CHR$ ... 7
- FILL$ ... 23
- INSTR ... 29
- LEN ... 31

STRIP ... 56
- Streifenfarbe ... 56

T

Tabellen ... 34
TAN ... 56
- Tangens ... 56

THEN ... 26
Töne ...
- BEEP ... 4
- BEEPING ... 5

TRA ... 57
- Zeichensatz ... 57

TURN ... 57
TURNTO ... 57
Turtle-Grafik ...
- FILL ... 23
- MOVE ... 37
- PENUP ... 43
- PENDOWN ... 43
- TURN ... 57
- TURNTO ... 57

U

Uhr ...
- ADATE ... 2
- DATE ... 14
- DATE$ ... 14
- DAY$ ... 14
- SDATE ... 42
- Überschreiben ... 40

UNDER ... 58
- Unterstreichen ... 48

V

Variabel
- CLEAR ... 8
- NEW ... 38

W

Wertzuweisung ... 32
WINDOW ... 59
- AT ... 3
- BLOCK ... 5
- BORDER ... 6
- CSIZE ... 12
- Cursor Kontrolle ... 3,12
- FILL ... 23
- FLASH ... 24
- INK ... 27

MODE 36
OVER 40
Unterstreichen 40
PAN 41
PAPER 41

Z

Zahlen 18
Zeichensatz 57
Zeilennummer 47
MERGE 35
GOSUB 39
GOTO 39
RUN 50
Zeit
Uhr stellen 2
Uhr zurücksetzen 52
Datum 14
PAUSE 42

sinclair

QL
Begriffe

In dem Abschnitt "Begriffe" werden die Begriffe im Zusammenhang mit SuperBASIC und der QL Hardware beschrieben. Es kann mit einem Nachschlagewerk verglichen werden. Ergeben sich aus der Benutzung des Computers oder der anderen Kapitel in diesem Handbuch Fragen im Zusammenhang mit SuperBASIC oder dem QL, so werden Sie die Antwort wahrscheinlich in diesem Abschnitt finden. Die Begriffe werden in alphabetischer Reihenfolge aufgeführt. Kann ein Themenkomplex nicht gefunden werden, so schlagen Sie bitte in dem Register nach, in dem angegeben wird, auf welcher Seite die gewünschte Information steht.

In den Fällen, in denen ein Beispiel mit Zeilennummern angeführt wird, handelt es sich um ein vollständiges Programm. Dieses Programm kann eingegeben und ausgeführt werden. Beispiele, die ohne Nummern aufgelistet werden, sind im allgemeinen einfache Befehle, die nicht unbedingt separat in den Computer eingegeben werden sollten.

by Stephen Berry *(Sinclair Research Limited)*

ANWEISUNG

Eine SuperBASIC-Anweisung ist ein Befehl an den QL zur Ausführung einer bestimmten Operation, z. B.:

```
LET a = 2
```

weist der mit a gekennzeichneten Variablen den Wert 2 zu.

In eine einzige Zeile kann mehr als eine Anweisung geschrieben werden, wobei die einzelnen Anweisungen durch einen Doppelpunkt (:) voneinander getrennt werden müssen. Zum Beispiel:

```
LET a = a + 2: PRINT a
```

addiert 2 zu dem Wert der Variablen **a** und speichert das Ergebnis wieder in **a**. Das Ergebnis wird dann ausgedruckt.

Steht vor einer Zeile keine *Zeilennummer,* so handelt es sich bei der Zeile um einen *direkten Befehl.* In diesem Fall verarbeitet SuperBASIC die Anweisung sofort. Steht vor der Anweisung eine Zeilennummer, so wird die Anweisung Bestandteil eines SuperBASIC-*Programms* und wird erst später ausgeführt.

Bestimmte SuperBASIC-Anweisungen wirken sich unter Umständen auf die anderen Anweisungen auf derselben Zeile aus, d. h. **IF, FOR, REPeat, REM** usw. Bestimmte SuperBASIC-Anweisungen sind nicht sinnvoll, wenn sie als direkte Befehle eingegeben werden.

AUFTEILUNG

Unter bestimmten Umständen muß man sich auf mehr als ein Element in einer Tabelle beziehen, d. h. die Tabelle aufteilen. Die Aufteilung einer Tabelle kann mit der Definition einer **Teiltabelle** oder einer Folge von Teiltabellen für SuperBASIC verglichen werden. Mit jedem Teil kann eine fortlaufende Folge von Elementen definiert werden, die zu einer bestimmten Dimension der Originaltabelle gehören. In diesem Zusammenhang bezieht sich der Ausdruck Tabelle auf eine numerische Tabelle, eine String-Tabelle oder einen einfachen String.

Es braucht kein Index für alle Dimensionen einer Tabell angegeben zu werden. Wird eine Dimension weggelassen, so werden Teile hinzugefügt, die den vollständigen Bereich von Elementen für diese bestimmte Dimension auswählen, d. h. den Teil (0 TO). SuperBASIC kann nur am Ende einer Liste mit Tabellenindizes Teile hinzufügen.

Syntax: *Index:=* | *numerischer_Ausdr* {einzelnes Element}
| *numerischer_Ausdr* TO *numerischer_Ausdr* {Bereich mit Elementen}
| *numerischer_Ausdr* TO {Bereich bis Ende}
| TO *numerischer_Ausdruck* {Bereich ab Anfang}

Tabellen_Referen:= | *Variable*
| *Variable* ([*Index* *[,*Index*]*])

Ein Tabellenteil kann in einer Zuweisung links oder rechts neben den " = " stehen.

Beispiel:

a) `PRINT daten_tabelle`
b) `PRINT buchstaben$(1 TO 15)`
c) `PRINT zwei_d_tabelle (3)(2 TO 4)`

Die Aufteilung von Strings wird auf dieselbe Art und Weise ausgeführt, wie die Aufteilung von numerischen Tabellen oder String-Tabellen.

a$(n)	Wählt das n-te Zeichen.
a$(n TO m)	Wählt sämtliche Zeichen von dem n-ten bis zum m-ten Zeichen einschließlich.
a$(n TO)	Wählt von einem Zeichen n bis zum Ende einschließlich.
a$(1 TO m)	Wählt vom Anfang bis zum m-ten Zeichen einschließlich. (Hier reicht nicht a$(TO m)).
a$	Wählt den ganzen Inhalt von a$.

Einige **BASIC**-Versionen verfügen über Funktionen namens **LEFT$**, **MID$**, **RIGHT$**.Sie sind in SuperBASIC nicht erforderlich. Die folgende Tabelle zeigt entsprechende SuperBASIC-Möglichkeiten:

SuperBASIC	Andere BASIC-Versionen
a$(n)	MID$ (a$,n,1)
a$(n TO m)	MID$ (a$,n,m + 1-n)
a$(1 TO n)	LEFT$ (a$,n)
a$(n TO)	RIGHT$ (a$,LEN(a$) + 1-n)

Hinweis

Werden Daten an eine aufgeteilte String-Tabelle oder String-Variablen zugewiesen, so kann dies unter Umständen nicht zu dem gewünschten Ergebnis führen. Bei auf diese Art und Weise vorgenommenen Zuweisungen wird die Länge des Strings nicht aktualisiert. Die Länge einer String-Tabelle oder String-Variablen wird nur aktualisiert, wenn dem ganzen String ein Wert zugewiesen wird.

AUSDRÜCKE

Bei SuperBASIC Ausdrücken kann es sich um *String-Ausdrücke, numerische Ausdrücke, bedingte Ausdrücke* oder eine Mischung dieser Ausdrücke handeln. Unvereinbare Datentypen werden automatisch vom System, wann immer möglich, geeignet umgewandelt.

Definition

monop:= | +
| –
| NOT

Ausdruck: = | [*monop*] *Ausdruck Operator Ausdruck*
| (*Ausdruck*)
| *Atom*

Atom:= | *Variable*
| *Konstante*
| *Funktion* [(*Audruck**[, *Ausdruck*]*)]
| *Tabellen_Element*

Variable:= | *Name*
| *Name%*
| *Name***$**

Funktion:= | *Name*
| *Name%*
| *Name***$**

Konstante:= | *Ziffer** [*Ziffer*]*
| *[*Ziffer*]*.*[*Ziffer*]*
| *[*Ziffer*]*[.]* [*Ziffer*] *N* [*Ziffer*]*

Die Auswertung des Ausdrucks kann eine ganze Zahl ergeben, die einen **ganzzahligen_Ausdruck** darstellt, einen String, der einen **String_Ausdruck** darstellt oder eine Gleitkommazahl, die einen **Gleitkomma_Ausdruck** darstellt. Häufig sind Gleitkomma- und ganzzahlige Ausdrücke gleichartig. In diesem Fall wird der Begriff **numerischer_Ausdruck** benutzt.

In einem Ausdruck können logische Operatoren benutzt werden. Ist die angegebene Operation wahr, so wird 1 als Ergebnis zurückgegeben. Ist die Operation falsch, so wird eine 0 zurückgegeben. Auch wenn die logischen Operatoren in einem beliebigen Ausdruck benutzt werden können, werden sie im allgemeinen in einer **IF**-Anweisung benutzt.

Beispiel:
a) `test_daten + 23.3 + 5`
b) `''abcdefghijklmnopqrstuvwxyz'' (2 TO 4)`
c) `32.1 * (farbe = 1)`
d) `zahl=-grenze`

BASIC

SuperBASIC enthält die meisten Funktionen, Prozeduren und Befehle, die auch in anderen BASIC-Versionen vorhanden sind. Einige dieser Funktionen sind in SuperBASIC überflüssig, wurden jedoch aus Gründen der Kompatibilität aufgenommen:

GOTO	Benutzen Sie **IF**, **REPeat** usw.
GOSUB	Benutzen Sie **DEFine PROCedure**
ON GOTO	Benutzen Sie **SELect**
ON GOSUB	Benutzen Sie **SELect**

Einige Befehle scheinen nicht vorhanden zu sein. Sie können jedoch stets über allgemeine Befehle benutzt werden. So sind beispielsweise in SuperBASIC keine **LPRINT**- oder **LLIST**-Befehle vorhanden. Die Ausgabe kann jedoch an einen Drukker geleitet werden, indem der entsprechende *Kanal* geöffnet und **PRINT** oder **LIST**-benutzt wird.

LPRINT	Benutzen Sie **PRINT #**
LLIST	Benutzen Sie **LIST #**
VAL	In SuperBASIC nicht erforderlich
STR$	In SuperBASIC nicht erforderlich
IN	Für 68008 nicht anwendbar
OUT	Für 68008 nicht anwendbar

Kommentar

Bei nahezu allen **BASIC**-Dialekten sind die Funktionen **VAL(x$)** und **STR$(x)** erforderlich, um den internen Code eines String-Ausdrucks in den internen Code eines numerischen Ausdrucks umwandeln zu können.

Diese Funktionen sind bei SuperBASIC überflüssig, weil es die Datentypumwandlung, wenn möglich, automatisch ausführt.

Die SuperBASIC-Befehle werden in dem Abschnitt *Befehle* im einzelnen definiert. Befehle haben dieselbe Form wie ein SuperBASIC Standard-*Name*. Es spielt keine Rolle, ob der Befehl mit Groß- oder Kleinbuchstaben geschrieben wird. Befehle werden in einer Mischung aus Groß- und Kleinbuchstaben aufgelistet und werden stets vollständig ausgeschrieben angezeigt. Einige Befehle können abgekürzt eingegeben werden. Der in Großbuchstaben angezeigte Teil stellt das Minimum dar, das mindestens eingegeben werden muß, damit SuperBASIC den Befehl erkennen kann.

Der SuperBASIC-Befehlsvorrat kann erweitert werden, indem *Prozeduren* hinzugefügt werden. Es wird empfohlen, die Namen dieser Prozeduren in Großbuchstaben anzugeben. So definierte Prozedurnamen werden stets von SuperBASIC in Großbuchstaben wiedergegeben. Auf diese Weise wird ihre Sonderfunktion in dem SuperBASIC-System kenntlich. Dagegen sollten die Namen von normalen Prozeduren in Kleinbuchstaben definiert werden.

Hinweis

Bestehende Befehle dürfen nicht als normale Namen innerhalb eines SuperBASIC-Programms benutzt werden. Nachfolgend werden die SuperBASIC-Befehle aufgeführt:

Befehlsliste			
ABS	DEFine PROCedure,	LEN	RANDOMISE
ACOS, ASIN,	END DEFine	LET	RND
ACOT, ATAN	DEG	LIST	RECOL
ADATE	DELETE	LOAD	REMark
ARC, ARC_R	DIM	LOCal	RENUM
AT	DIMN	LN, LOG10	REPeat,
AUTO	DIR	LRUN	END REPeat
BAUD	DIV	MERGE	RESPR
BEEP	DLINE	MOD	RETurn
BEEPING	EDIT	MODE	RETRY
BLOCK	ELLIPSE,	MOVE	RUN
BORDER	ELLIPSE_R	MRUN	SAVE
CALL	EOF	NET	SIN
CHR$	EXEC, EXEC_W	NEW	SCALE
CIRCLE,	EXIT	NEXT	SCROLL
CIRCLE_R	EXP	ON GO TO,	SDATE
CLEAR	FILL	ON GO SUB	SELect,
CLOSE	FILL$	OPEN, OPEN_IN,	END SELect
CLS	FLASH	OPEN_NEW	SEXEC
CODE	FOR,	OVER	SQRT
CONTINUE,	END FOR	PAN	STOP
RETRY	FORMAT	PAPER	STRIP
COPY, COPY_N	GO SUB,	PAUSE	TAN
COS	GO TO	PEEK, PEEK_W,	TO
COT	IF, THEN, ELSE,	PEEK_L	TURN,
CSIZE	END IF	PENUP	TURN TO
CURSOR	INK	PENDOWN	UNDER
DATA, READ,	INKEY$	PI	VER$
RESTORE	INPUT	POINT, POINT_R	WIDTH
DATE$, DATE	INSTR	POKE, POKE_W,	WINDOW
DAY$	INT	POKE_L	
DEFine FuNction,	KEYROW	PRINT	
END DEFine	LBYTES	RAD	

BILDSCHIRM

Modus 512 Der Bildschirm hat eine Größe von 512 x 256 (horizontal x vertikal) Pixeln. Von den Vollfarben können nur:

Schwarz
Rot
Grün
Weiß

in dieser Betriebsart angezeigt werden.

Modus 256 Der Bildschirm hat eine Größe von 256 x 256 (horizontal x vertikal) Pixeln. In dieser Betriebsart können sämtliche Vollfarben benutzt werden:

Blau
Rot
Magenta
Grün
Cyan
Gelb
Weiß

Bei der niedrigauflösenden Betriebsart ist auch ein Blinken vorgesehen.

Hinweis Bei einem normalen Fernsehgerät kann der vollständige QL-Bildschirm nicht angezeigt werden. Teile der Bildschirmanzeige am oberen Ende und auf den Seiten werden nicht wiedergegeben. In dem Standardfenster wird diese Tatsache berücksichtigt und die tatsächliche Bildgröße verkleinert. Die volle Größe kann mit dem **WINDOW**-Befehl hergestellt werden.

Befehl	Funktion
MODE	Legt den Bildschirm Modus fest

DATEITYPEN
DATEIEN

Die gesamte Ein-/Ausgabe bei dem QL wird von einer bzw. zu einer *logischen Datei* vorgenommen. Es gibt verschiedene Dateitypen:

Daten

SuperBASIC-Programme, Text-Dateien. Sie werden mit **PRINT, SAVE** erstellt. Der Zugriff erfolgt über **INPUT, INKEY$, LOAD** usw.

Exec

Ein ausführbares transientes Programm. Diese Dateitypen werden mit **SEXEC** gesichert und mit **EXEC, EXEC_W** usw. geladen.

Code

Unformatierte Daten im Speicher, Bildschirmabbilder usw. Sie werden mit **SBYTES** gesichert und mit **LBYTES** geladen.

DATENTYPEN VARIABLE

Ganze Zahlen

Ganze Zahlen können im Bereich von −32768 bis +32767 dargestellt werden. *Variablen* werden als ganze Zahlen betrachtet, wenn hinter dem Variablennamen ein Prozentzeichen **%** steht. In SuperBASIC gibt es keine ganzzahligen Konstanten, so daß alle Konstanten als *Gleitkomma*-Zahlen gespeichert werden.

Syntax: *Name***%**

Beispiel:
a) `zahl%`
b) `max_groß%`
c) `das_ist_eine_ganzzahlige_variable%`

Gleitkommazahlen

Gleitkommazahlen können im Bereich von +/− (10^{-615} bis 10^{+615}) mit acht signifikante Ziffern dargestellt werden. Die Gleitkommazahl ist der Standard-Datentyp bei SuperBASIC. Sämtliche Konstanten werden in Gleitkommaform gespeichert und können in Exponential-Schreibweise eingegeben werden.

Syntax: *Name* | *Konstante*

Beispiel:
a) `momentaner_wert`
b) `76.2356`
c) `354E25`

String

Ein String ist eine Zeichenfolge aus maximal 32766 Zeichen. Steht hinter dem Variablennamen ein **$**, so handelt es sich um eine *Variable* vom Typ String. String-Daten werden dargestellt, indem die zu dem String gehörenden Zeichen entweder zwischen Apostrophe oder zwischen Anführungszeichen gesetzt werden.

Syntax: *Name$* | *"Text"*

Beispiel:
a) `string_variable$`
b) `"Das sind String - Daten"`
c) `"Das ist noch ein String"`

Name

Namen dienen dem System dazu, *Variable, Prozeduren, Dateien* usw. zu identifizieren.

Syntax: *Name*

Beispiel:
a) `mdv1_daten_datei`
b) `ser1_`

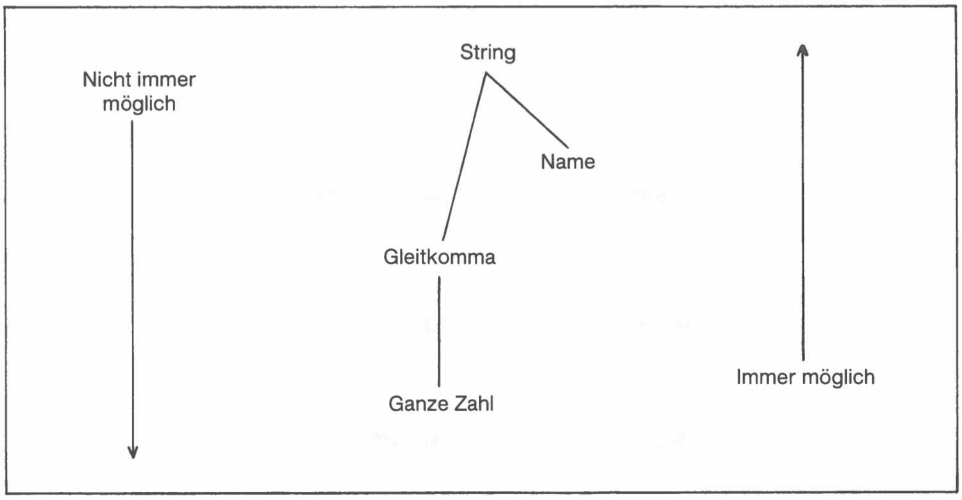
DATENTYP-UMWANDLUNG

Falls erforderlich wandelt SuperBASIC einen nicht passenden Datentyp in einen Typ um, mit dem die angegebene Operation ausgeführt werden kann.

Mit den benutzten *Operatoren* wird die erforderliche Umwandlung festgelegt. Erfordert eine Operation beispielsweise einen *String-Parameter* und wird ein *numerischer* Parameter angegeben, so wandelt SuperBASIC den Parameter als erstes in einen String-Parameter um. Nun können allerdings die Daten nicht immer in den erforderlichen Typ umgesetzt werden. In diesem Fall wird eine Fehlermeldung ausgegeben.

Der Typ eines *Funktions-* oder *Prozedur*-Parameters kann ebenfalls in den richtigen Typ umgewandelt werden. So erfordert der SuperBASIC-Befehl **LOAD** beispielsweise einen Parameter vom Typ *Namen,* kann jedoch einen Parameter vom Typ *String* akzeptieren, der von der Prozedur selbst in den richtigen Typ umgesetzt wird.

Bei dem QL gibt es eine natürliche Reihenfolge der Datentypen. (Siehe Abbildung.) *String* ist der allgemeinste Typ, da mit ihm *Namen*, *Gleitkommazahlen* und *ganze Zahlen* dargestellt werden können. Der *Gleitkommatyp* ist nicht so allgemein wie String, ist jedoch allgemeiner als ganze Zahl, da Gleitkommazahlen ganzzahlige Werte (meistens genau) darstellen können. In der folgenden Abbildung wird die Reihenfolge in Form eines Diagramms dargestellt. Die Daten können in dem Programm stets in Aufwärtsrichtung, jedoch nicht immer in Abwärtsrichtung umgewandelt werden.



Beispiel **a = b + c**		(Eine Umwandlung ist vor der Addition nicht erforderlich. Eine Umwandlung ist nicht erforderlich vor der Zuweisung des Ergebenisses an a.)
	a% = b + c	(Eine Umwandlung ist vor der Addition nicht erforderlich. Das Ergebnis wird jedoch vor der Zuweisung in eine ganze Zahl umgewandelt.)
	a$ = b$ + c$	(**b$** und **c$** werden wenn möglich in die Gleitkommaform umgewandelt, bevor sie addiert werden. Das Ergebnis wird vor der Zuweisung in einen String umgewandelt.)
	LOAD "mdv1_data"	(Der String **"mdv1_data"** wird von der Lade-Prozedur vor der Benutzung in einen Namen umgewandelt.)

Kommentar

Mit SuperBASIC können Anweisungen geschrieben werden, die in den meisten anderen Computersprachen zu Fehlermeldungen führen. Im allgemeinen können Datentypen sehr flexibel gemischt werden:

a) `PRINT "1" + 2 + "3"`
b) `LET a$ = 1 + 2 + a$ + "4"`

DATEN-ÜBERTRAGUNG RS-232-C

Der QL verfügt über zwei serielle Anschlüsse (mit der Bezeichnung SER1 und SER2), über die er an Anlagen angeschlossen werden kann, die den EIA-Standard RS-232-C oder einen kompatiblen Standard benutzen.

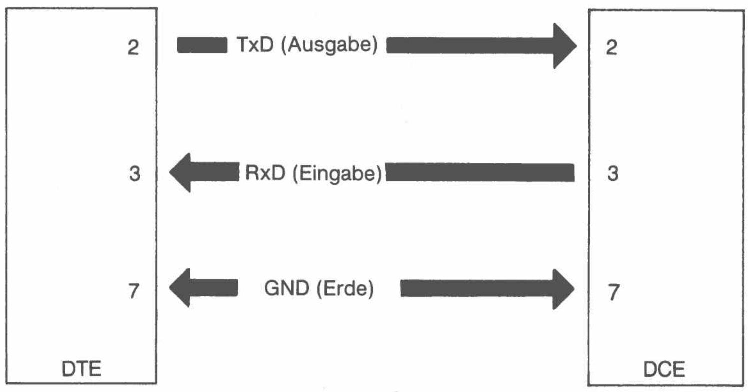
Der RS-232-C Standard war ursprünglich für das Senden und Empfangen von Daten über Telefonleitungen mit Hilfe eines Modems gedacht. Nun wird er jedoch häufig für den direkten Anschluß von Computern untereinander und für den Anschluß an verschiedene Peripheriegeräte, z.B. Drucker, Plotter usw., benutzt.

Da der RS-232-C Standard in vielen verschiedenen Formen in verschiedenen Geräteteilen benutzt wird, kann es selbst für einen Fachmann äußerst schwierig sein, zwei Geräteteile, bei denen es sich angeblich um RS-232-C Standardanschlüsse handelt, miteinander zu verbinden. In diesem Abschnitt sollen die meisten der grundlegenden Probleme erläutert werden, die hierbei auftreten können.

Der RS-232-C Standard bezieht sich auf zwei Arten von Einrichtungen:

1. Datenendeinrichtungen (DTE)
2. Datenübertragungseinrichtungen (DCE)

Bei dem Standard wurde davon ausgegangen, daß das Terminal (im allgemeinen die Datenendeinrichtung) und das Modem (im allgemeinen die Datenübertragungseinrichtung) beide über dieselbe Art von Anschluß verfügen.



In dem obigen Diagramm wird dargestellt, wie das DTE (die Datenendeinrichtung) Daten an Steckerstift 2 überträgt, während das DCE (die Datenübertragungseinrichtung) Daten über ihren Steckerstift 2 empfangen muß. (Die Daten werden dennoch als Sendedaten bezeichnet!) Gleichermaßen empfängt das DTE Daten über Steckerstift 3, während das DCE Daten über Steckerstift 3 sendet. (Sie werden dennoch als Empfangsdaten bezeichnet!) Dies an sich ist schon verwirrend, kann jedoch noch zu weit größeren Problemen führen, wenn man sich nicht einig ist, ob eine bestimmte Einheit als Datenübertragungseinrichtung oder Datenendeinrichtung geschaltet werden soll.

Bedauerlicherweise entscheiden sich einige Benutzer, daß ihre Computer als Datenübertragungseinrichtung gestaltet werden, während andere ähnliche Computer als Datenendeinrichtung gestalten. Dies führt natürlich zu Schwierigkeiten bei der Benutzung der seriellen Anschlüsse bei den einzelnen Anlagenteilen.

Bei dem QL ist SER1 als Datenübertragungseinrichtung (DCE) gestaltet, während SER2 als Datenendeinrichtung (DTE) gestaltet ist. Demzufolge muß es möglich sein, zumindest einen der seriellen Anschlüsse mit einer bestimmten Einheit zu verbinden, indem einfach der Anschluß benutzt wird, der "richtig" verdrahtet ist. Die Belegung der Anschlußstifte wird nachfolgend angegeben. Sinclair Research Limited kann ein Kabel für den Anschluß des QL an einem 25poligen D-Standardstecker liefern.

SER1				SER2		
Stecker-stift	Name	Funktion		Stecker-stift	Name	Funktion
1	GND	Signal-Erde		1	GND	Signal-Erde
2	TxD	Senden		2	TxD	Empfangen
3	RxD	Empfangen		3	RxD	Senden
4	DTR	Empfangsbereit-schaft		4	DTR	Sendebereit-schaft
5	CTS	Sendebereit-schaft		5	CTS	Empfangsbe-reitschaft
6,7,8	–	0–Volt		6,7,8	–	0–Volt
9	–	+12 V		9	–	+12 V

TxD Sendedaten
RxD Empfangsdaten

DTR Datenendeinrichtung betriebsbereit
CTS Sendebereitschaft

Nachdem das Gerät an den "richtigen" Anschluß angeschlossen wurde, muß die Baudrate (die Geschwindigkeit für die Datenübertragung) so eingestellt werden, daß die Baudraten für den QL und die angeschlossene Einheit identisch sind. Der QL kann für folgende Übertragungsgeschwindigkeiten eingestellt werden:

75
300
600
1200
2400
4800
9600
19200 Baud (nur Senden)

Die Baudrate des QL wird mit dem **BAUD**-Befehl festgelegt. Die Baudrate ist für beide seriellen Anschlüsse immer gleich.

Die von dem QL benutzte **Parität** muß ebenfalls so gewählt werden, daß sie mit der Parität des Peripheriegerätes übereinstimmt. Mit der Paritätsprüfung werden normalerweise einfache Übertragungsfehler entdeckt. Hier kann eine gerade, ungerade, Mark-, Space- oder keine Parität festgelegt werden. Im letzteren Fall bedeutet dies, daß alle acht Bits des Bytes für die Daten benutzt werden.

Mit den **Stoppbits** wird das Ende der Übertragung eines Bytes oder Zeichens angegeben. Der QL empfängt Daten mit einem, eineinhalb oder zwei Stoppbits. Die Daten werden stets mit mindestens zwei Stoppbits gesendet. Ist der QL auf 9600 Baud eingestellt, so kann er keine Daten mit nur einem Stoppbit empfangen. In diesem Fall sind mindestens eineinhalb Stoppbits erforderlich.

Unter Umständen sind zwischen dem QL und einem angeschlossenen Gerät **Handshake**-Leitungen erforderlich. Über diese Leitungen können der QL und das Peripheriegerät die Geschwindigkeit der Datenübertragung überwachen und kontrollieren. Dies ist unter Umständen erforderlich, wenn eine der beiden Einheiten die Daten nicht mit der Geschwindigkeit verarbeiten kann, mit der sie übertragen werden. Der QL benutzt zwei Handshake-Leitungen:

CTS Sendebereitschaft
DTR Datenendeinrichtung betriebsbereit.

Kann die Datenendeinrichtung die Daten nicht in der übertragenen Geschwindigkeit verarbeiten, so kann sie die DTR-Leitung abschalten. Dadurch weiß die Datenübertragungseinrichtung, daß sie das Senden von Daten stoppen muß. Sobald die Datenendeinrichtung die empfangenen Daten verarbeitet hat, teilt sie der Datenübertragungseinrichtung über die DTR-Leitung mit, daß die Übertragung wieder aufgenommen werden kann. Auf dieselbe Art und Weise kann die Datenübertragungseinrichtung das Senden von Daten durch die Datenendeinrichtung stoppen, indem sie die CTS-Leitung abschaltet. Sind zusätzliche Steuersignale erforderlich, so können sie mit der 12 V Stromversorgung verdrahtet werden, die bei beiden seriellen Anschlüssen zur Verfügung steht.

Auch wenn die Übertragung von dem QL häufig ohne Handshaking möglich ist, kann der QL unter keinen Umständen Daten korrekt ohne Anschluß der CTS-Leitung bei SER1 und DTR bei SER2 empfangen.

Die Datenübertragung bei dem QL wird im Vollduplex-Modus vorgenommen, d.h. das Senden und Empfangen kann gleichzeitig erfolgen.

Parität und Handshaking werden ausgewählt, sobald der serielle Kanal geöffnet wird.

Befehl	Funktion
BAUD	Legt die Übertragungsgeschwindigkeit fest
OPEN	Öffnet die seriellen Kanäle*
CLOSE	Schließt die seriellen Kanäle

* Für eine genaue Beschreibung wird auf den Begriff *Einheit* verwiesen.

DIREKTER BEFEHL

SuperBASIC unterscheidet zwischen einer Anweisung, vor der eine Zeilennummer eingegeben wird, und einer Anweisung, die ohne eine Zeilennummer eingegeben wird. Ohne Zeilennummer handelt es sich bei der Anweisung um einen **direkten Befehl.** Er wird unmittelbar von dem SuperBASIC **Befehls-Interpreter** verarbeitet. Wird beispielsweise **RUN** in die Befehlszeile eingegeben, dann wird ein gespeichertes Programm gestartet. Wird eine Anweisung mit einer Zeilennummer eingegeben, so wird die Syntax der Zeile geprüft. Entdeckte Syntaxfehler werden angegeben. Eine richtige Zeile wird in das SuperBASIC Programm übernommen und gespeichert. Diese Anweisungen stellen ein SuperBASIC-*Programm* dar und werden nur ausgeführt, wenn das Programm mit dem **RUN**- oder **GOTO**-Befehl gestartet wird.

Nicht alle SuperBASIC-Anweisungen können sinnvoll als direkte Befehle eingegeben werden. Dies gilt beispielsweise für **END FOR, END DEFine** usw.

EINHEITEN

Die Einrichtungen, zu denen der QL Daten senden oder von denen er Daten empfangen kann, heißen **Einheiten.**

Das System legt nicht von vornherein fest, welche Einheiten schließlich zur Laufzeit eines Programms verwendet wird. Die Einheit kann deshalb leicht geändert werden. Es ist auch möglich, Daten zu einer anderen als der ursprünglich vorgesehenen Einheit umzuleiten. Zum Beispiel muß vielleicht ein *Programm* irgendwann während seiner Laufzeit Daten an einen Drucker ausgeben. Wenn der Drucker gerade nicht verfügbar ist, dann können die Daten zu einer *Microdrive-Datei* umgeleitet und dort gespeichert werden. Die Datei kann dann zu einem späteren Zeitpunkt ausgedruckt werden. E/A beim QL kann aufgefaßt werden als Lesen von oder Schreiben zu einer **logischen Datei,** die sich in einer einheitenunabhängigen Standardform befindet.

Sämtliche einheitenabhängigen Operationen werden von einzelnen **Einheitentreibern** ausgeführt, die speziell für jede QL-Einheit geschrieben wurden. Das System kann automatisch Treiber für Peripheriegeräte suchen und einsetzen. Sie müssen in dem Standardformat für QL-Einheitentreiber geschrieben werden. (Siehe Begriff *Peripherie-Erweiterung.*)

Wird eine Einheit aktiviert, so wird ein *Kanal* geöffnet und mit der Einheit verknüpft. Um einen Kanal richtig zu öffnen, müssen gelegentlich grundlegende Informationen über die Einheit geliefert werden. Diese zusätzliche Information wird an den Einheitennamen angehängt.

Der Dateiname muß den Regeln für SuperBASIC-*Namen* entsprechen. Ein Dateiname (Einheitenname) kann auch als SuperBASIC-*String-Ausdruck* angegeben werden.

Allgemein hat ein Dateiname folgende Form:

Name [*Informationen*]

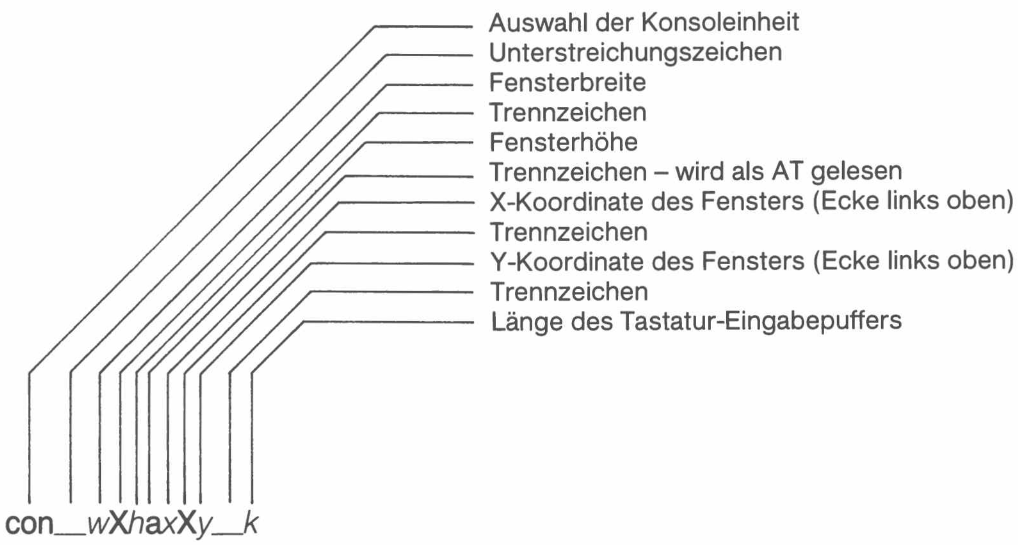
Jede logische Einheit in dem System erfordert ihre eigenen besonderen "Zusatzinformationen", auch wenn wo immer möglich Standardwerte für die Parameter eingesetzt werden.

Definition

Einheit:= Name

Die Form des Einheitennamens wird nachfolgend dargelegt.

Beispiel: Für Konsoleinheiten



CON_*w*X*h*a*x*X*y*_*k*

Konsol-E/A

[*w*X*h*]	– *Breite, Höhe* des Fensters
[a*x*X*y*]	– *X,Y*-Koordinaten des Fensters (Ecke links oben)
[*k*]	– Länge des Tastatur-Eingabepuffers (Bytes)

Standardwert: `con_448x180a32x16_128`

Beispiel: a) `OPEN #4, con_20x50a0x0_32`
b) `OPEN #8, con_20x50`
c) `OPEN #7, con_20x50a10x10`

SCR_*w*X*h*a*x*X*y*

Bildschirmausgabe

[wX*h*] – *Breite, Höhe* des Fensters
[axXy] – *X,Y*-Koordinaten des Fensters (Ecke links oben).

Standardwert: scr_448x180a32x16

Beispiel: a) `OPEN #4, scr_10x10a20x50`
b) `OPEN #5, scr_10x10`

SER*nphz*

Serieller Anschluß (RS-232-C)
n Anschluß-Nummer (1 oder 2)

[*p*] Parität	[*h*] Handshaking	[*z*] Protokoll
e – Gerade	i – Ignorieren	r – Unformatierte Daten, kein Dateiende
o – Ungerade	h – Handshake	z – CTRL Z (CHR$(26)) entspricht Dateiende
m – Mark		c – Wie z, wandelt jedoch ASCII 10 (Qdos Zeichen für Zeilenvorschub) in ASCII 13 (<CR>) um
s – Space		

Standardwert: ser1rh (8 Bit, keine Parität, mit Handshake)

Beispiel: a) `OPEN #3, ser1e`
b) `OPEN #4, serc`
c) `COPY mdv1_test_datei TO ser1c`

NET*d*_*s*

Serielle Netzwerk-E/A

[*d*] gibt die Richtung an
i – Eingabe
o – Ausgabe

[s] Stationsnummer
0 – Senden an alle
eigene Station – für allgemeinen Empfang (nur Eingabe)

Standardwert: kein Standardwert

Beispiel: a) `OPEN #7, neti_32`
b) `OPEN #4, neto_0`
c) `COPY ser1 TO neto_0`

MDV*n*_*name*

Microdrive-Dateizugriff
n Microdrive-Nummer
name Microdrive-Dateiname

Standardwert: kein Standardwert

Beispiel: a) `OPEN #9, mdv1_daten_datei`
b) `OPEN #9, mdv1_test_programm`
c) `COPY mdv1_test_datei TO scr`

Befehl	Funktion
OPEN	Eröffnet einen Kanal zu einer Einheit und aktiviert sie zur Benutzung
CLOSE	Schließt einen Kanal
COPY	Kopiert Daten zwischen den Einheiten
COPY_N	Kopiert Daten zwischen den Einheiten, allerdings wird die Information aus den Dateikennsätzen nicht kopiert
EOF	Test auf Dateiende
WIDTH	Legt die Breite fest

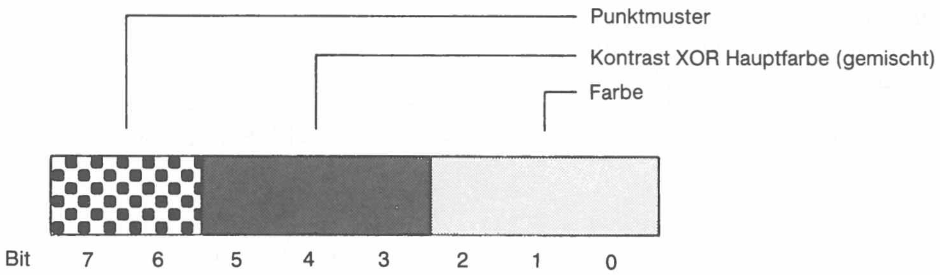
FARBE

Bei dem QL können entweder **Vollfarben** oder **Punktmuster** benutzt werden – eine Mischung zweier Farben mit einem bestimmten Muster. Die Farbangabe bei dem QL kann maximal drei Parameter umfassen: eine Farbe, eine Kontrastfarbe und ein Punktmuster.

Ein Parameter

Farbe:= zusammengesetzte_Farbe

Mit dem einzelnen Argument werden die drei Teile der Farbspezifikation angegeben. Die Hauptfarbe steht in den drei niederwertigen Bits des Farb-Bytes. Die nächsten drei Bits enthalten das Exklusive Oder (XOR) der Hauptfarbe und die Kontrastfarbe. Die beiden höchstwertigen Bits geben das *Punkt-Muster* an.



Werden nur die drei niederwertigen Bits (d. h. die erforderliche Farbe) angegeben, so wird kein *Punktmuster* angefordert und eine einzige Vollfarbe angezeigt.

Zwei Parameter

Farbe:= Hintergrund, Kontrast

Bei der *Farbe* handelt es sich um ein *Punktmuster* aus den beiden angegebenen Farben. Hier wird von dem Standard-Punktmuster (Punktmuster 3) ausgegangen.

Drei Parameter

Farbe:= Hintergrund, Kontrast, Punktmuster

Die *Hintergrund*- und *Kontrast-Farben* und das *Punktmuster* werden jeweils separat definiert.

Farben

Die Wirkung der Farbcodes ist in beiden Bildschirmbetriebsarten verschieden, da in der hochauflösenden Betriebsart weniger Farben dargestellt werden können.

Code	Bitmuster	Zusammensetzung	Farbe	
			8 Farben	4 Farben
0	0 0 0		Schwarz	Schwarz
1	0 0 1	Blau	Blau	Schwarz
2	0 1 0	Rot	Rot	Rot
3	0 1 1	Rot + Blau	Magenta	Rot
4	1 0 0	Grün	Grün	Grün
5	1 0 1	Grün + Blau	Zyan	Grün
6	1 1 0	Grün + Rot	Gelb	Weiß
7	1 1 1	Grün + Rot + Blau	Weiß	Weiß

Farbzusammensetzung und Codes

Punktmuster

Bei Punktmustern werden eine Hintergrund- und eine Kontrast-Farbe zu einem feinen Punktmuster gemischt. Punktmuster können bei dem QL wie normale Vollfarben benutzt werden, auch wenn die Punktmuster auf einem normalen Fernsehbildschirm nicht richtig wiedergegeben werden können. Es gibt vier verschiedene Punktmuster:



Punktmuster 0 Punktmuster 1 Punktmuster 2 Punktmuster 3

Punktmuster 3 ist der Standardwert.

Beispiel:
a) `PAPER 0,2,0: CLS`
b) `PAPER 2,5: CLS`
c) `PAPER 255: CLS`

Hinweis

Punktmuster werden unter Umständen auf einem Fernsehgerät, das über den UHF-Anschluß gespeist wird, nicht richtig wiedergegeben.

FEHLER-BEHANDLUNG

Fehler werden bei SuperBASIC in einem Standardformat gemeldet.

In Zeile *Zeilennummer Fehlertext*

wobei Zeilennummer angibt, in welcher Zeile der Fehler entdeckt wurde. Der Fehlertext wird daran anschließend aufgeführt.

(1) ABGEBROCHEN
Eine Operation wurde vorzeitig beendet (oder die Tastenkombination für die Unterbrechung wurde betätigt).

(2) FEHLERHAFTER JOB
Eine Fehlermeldung von Qdos, die sich auf Systemaufrufe für die Steuerung des Multitasking oder die E/A bezieht.

(3) SPEICHERÜBERLAUF
Qdos und/oder SuperBASIC verfügen nicht über ausreichend freien Speicherplatz.

(4) BEREICHSÜBERLAUF
Diese Fehlermeldung wird angezeigt, wenn versucht wird, außerhalb eines Fensters zu schreiben, oder wenn ein falscher Tabellenindex benutzt wird.

(5) PUFFER VOLL
Bei einer E/A-Operation war der Puffer gefüllt, bevor ein Satzendezeichen gefunden wurde.

(6) KANAL NICHT ERÖFFNET
Es wurde versucht, in einem Kanal zu lesen oder zu schreiben bzw. einen Kanal zu schließen, der nicht geöffnet war. Diese Fehlermeldung kann auch angezeigt werden, wenn ein Kanal nicht geöffnet werden kann.

(7) NICHT GEFUNDEN
Das Dateiverwaltungssystem, die Einheit, der Datenträger oder die Datei können nicht gefunden werden.

SuperBASIC kann einen Namen nicht finden. Diese Fehlermeldung kann bei falsch verschachtelten Strukturen angezeigt werden.

(8) EXISTIERT BEREITS
Das Dateiverwaltungssystem hat eine schon bestehende Datei gefunden, die denselben Namen hat, wie eine neue Ausgabedatei.

(9) IN BEARBEITUNG
Das Dateiverwaltungssystem hat eine Datei oder Einheit ermittelt, die schon exklusiv benutzt wird.

(10) DATEIENDE
Während der Eingabe wird das Dateiende entdeckt.

(11) DATENTRÄGER VOLL
Eine Einheit ist schon voll (im allgemeinen Microdrive).

(12) UNGÜLTIGE BEZEICHNUNG
Das Dateiverwaltungssystem hat den Namen erkannt, es ist jedoch ein Syntaxfehler oder ein Fehler im Parameterwert vorhanden.

Bei SuperBASIC bedeutet dies, daß ein Name im falschen Kontext benutzt wurde. So wurde beispielsweise eine Variable als Prozedur benutzt.

(13) ÜBERTRAGUNGSFEHLER
RS-232-C-Paritätsfehler.

(14) FORMATFEHLER
Ein Datenträger konnte nicht formatiert werden. Der Datenträger ist möglicherweise fehlerhaft (im allgemeinen eine Microdrive-Kassette).

(15) UNGÜLTIGER PARAMETER
In der Parameterliste eines Befehls, einer SuperBASIC-Prozedur oder eines Funktionsaufrufes ist ein Fehler vorhanden.

Es wurde versucht, Daten von einer Nur-Schreib-Einheit zu lesen.

(16) FEHLERHAFTER DATENTRÄGER
Der Datenträger (im allgemeinen eine Microdrive-Kassette) ist wahrscheinlich fehlerhaft.

(17) FEHLER IM AUSDRUCK
Bei der Auswertung eines Ausdrucks wurde ein Fehler entdeckt.

(18) ÜBERLAUF
Arithmetischer Überlauf, Division durch Null, Quadratwurzel einer negativen Zahl usw.

(19) NICHT IMPLEMENTIERT

(20) NUR LESEN
Es wurde versucht, Daten in eine gemeinsam benutzte Datei zu schreiben.

(21) SYNTAXFEHLER
Ein SuperBASIC-Syntaxfehler wurde gefunden.

(22) PROC/FN GELÖSCHT
Diese Meldung wird nur zur Information ausgegeben und gibt keinen Fehler an. Mit dieser Meldung wird darauf hingewiesen, daß das Programm gestoppt und nachfolgend geändert wurde, wobei SuperBASIC auf die äußere Programmebene zurückkehren mußte. Eventuell wirksame Prozeduraufrufe wurden außer Kraft gesetzt.

Fehlerbehebung

Nachdem es zu einem Fehler gekommen ist, kann das Programm bei der nächsten Anweisung durch Eingabe von

```
CONTINUE
```

fortgesetzt werden. Kann die Fehlerbedingung korrigiert werden, ohne das Programm zu ändern, so kann das Programm bei der Anweisung neu gestartet werden, die den Fehler ausgelöst hat. Hierzu wird:

```
RETRY
```

eingegeben.

FENSTER

Fenster sind *Bildschirm-Bereiche,* die sich in fast jeder Hinsicht so verhalten, als wäre jedes einzelne Fenster ein Bildschirm für sich, d. h. das Fenster wird gerollt, wenn es mit Text gefüllt ist, kann mit dem **CLS**-Befehl gelöscht werden usw.

Fenster können angegeben und mit einem Kanal verknüpft werden, wenn der Kanal *geöffnet* wird. Die aktuelle Fenster-Form kann mit dem **WINDOW**-Befehl geändert werden. Mit dem **BORDER**-Befehl kann ein Fenster eingerahmt werden. Die Ausgabe kann über den entsprechenden Kanal zu einem Fenster geleitet werden. Die Eingabe kann so geleitet werden, daß sie von einem bestimmten Fenster stammt, indem die Eingabe über den entsprechenden *Kanal* vorgenommen wird. Ist mehr als ein Kanal zur Eingabe bereit, so kann die Eingabe zwischen den bereiten Kanälen umgeschaltet werden, indem

[CTRL] C

betätigt wird. Der Cursor blinkt in dem ausgewählten Fenster auf.

Fenster können für grafische und nichtgrafische Ausgaben gleichzeitig benutzt werden. Die nichtgrafische Ausgabe bezieht sich auf die aktuelle Cursor-Position, die mit dem **CURSOR**-Befehl an eine beliebige Stelle innerhalb des angegebenen Fensters gesetzt werden kann. Mit dem **AT**-Befehl kann sie in jede beliebige Zeilen-/Spalten-Position gesetzt werden. Die Grafik-Ausgabe ist auf das Grafik-Koordinatensystem oder den Grafik-Cursor bezogen, der mit den *Grafik-Prozeduren* positioniert und bewegt werden kann.

Teile

Bestimmte Befehle (**CLS, PAN** usw.) können einen Parameter erhalten, mit dem ein Teil des aktuellen Fensters für ihre Operation definiert wird. Dieser Parameter entspricht der nachfolgenden Definition:

Teil	Beschreibung
0	Ganzer Bildschirm
1	Über der Cursor-Zeile, diese nicht eingeschlossen
2	Ende des Bildschirms, Cursor-Zeile nicht eingeschlossen
3	Ganze Cursor-Zeile
4	Zeile rechts vom Cursor, dieser eingeschlossen

Befehl	Funktion
WINDOW	Definiert ein Fenster neu
BORDER	Zeichnet einen Rahmen um ein Fenster
PAPER	Definiert die Hintergrundfarbe für ein Fenster
INK	Definiert die Schriftfarbe für ein Fenster
STRIP	Definiert eine Streifenfarbe für ein Fenster
PAN	Verschiebt den Inhalt eines Fensters waagerecht
SCROLL	Rollt den Inhalt eines Fensters
AT	Legt die Druckposition fest
CLS	Löscht ein Fenster
CSIZE	Legt die Zeichengröße fest
FLASH	Legt ein Blinken der Zeichen fest
RECOL	Legt eine neue Farbe für ein Fenster fest

FUNKTIONEN UND PROZEDUREN

SuperBASIC *Funktionen und Prozeduren* werden mit den Anweisungen **DEFine FuNction** und **DEFine PROCedure** definiert. Eine Funktion wird durch Eingabe ihres Namens in einem SuperBASIC-Ausdruck aufgerufen. Die Funktion muß in einem Ausdruck enthalten sein, da sie einen Wert liefert, der verarbeitet werden muß. Eine Prozedur wird aufgerufen, indem ihr Name als erstes Element in einer SuperBASIC-Anweisung eingegeben wird.

Daten können an eine Funktion oder Prozedur übergeben werden, indem im Anschluß an den Funktions- oder Prozedurnamen eine Liste mit **aktuellen Parametern** angehängt wird. Diese Liste wird mit einer entsprechenden Liste verglichen, die bei der Definition an den Namen der Funktion oder Prozedur angehängt wurde. Diese zweite Liste enthält die **formalen Parameter** der Funktion oder der Prozedur. Bei den formalen Parametern muß es sich um SuperBASIC-Variablen handeln. Bei den aktuellen Parametern muß es sich um eine *Tabelle,* einen *Tabellen-Teil* oder einen SuperBASIC-*Ausdruck* handeln, wobei eine einzelne *Variable* oder *Konstante* die einfachste Form eines Ausdrucks darstellt.

Mit den formalen Parametern wird nur angegeben, wie die aktuellen Parameter verarbeitet werden müssen. Die Elemente in jeder Parameterliste werden paarweise in der richtigen Reihenfolge zusammen mit der Funktion oder Prozedur aufgerufen. Die formalen Parameter werden dann gleichbedeutend mit den aktuellen Parametern. Es gibt drei verschiedene Möglichkeiten, Parameter zu benutzen.

Handelt es sich bei dem aktuellen Parameter um eine einzelne Variable und werden dem formalen Parameter in der Funktion oder Prozedur Daten zugewiesen, so werden die Daten auch dem entsprechenden aktuellen Parameter zugewiesen.

Handelt es sich bei dem aktuellen Parameter um einen Ausdruck, so hat die Zuweisung von Daten an den entsprechenden formalen Parameter keine Auswirkungen außerhalb der Prozedur. Eine Variable wird zu einem Ausdruck, indem sie in Klammern gesetzt wird.

Handelt es sich bei dem aktuellen Parameter um eine Variable, wurde ihr jedoch vorher kein Wert zugewiesen, so wird durch Wertzuweisung an den entsprechenden formalen Parameter die Variable besetzt, die als aktueller Parameter angegeben wurde.

Variablen können mit der Anweisung **LOCal** als lokal in einer Funktion oder Prozedur definiert werden. Lokale Variablen haben keine Auswirkung auf Variablen mit demselben Namen außerhalb der Funktion oder Prozedur, in der sie definiert sind. Auf diese Weise können bedeutungsvolle Variablennamen freier ausgewählt werden, ohne daß externe Variablen dadurch beeinflußt werden. Eine lokale Variable ist für jede Funktion oder Prozedur verfügbar, die aus der Prozedur oder Funktion aufgerufen wird, in der die Variable als lokale Variable deklariert wurde. Dies gilt nicht, wenn die aufgerufene Funktion oder Prozedur eine weitere lokale Variable mit demselben Namen enthält.

Funktionen und Prozeduren in SuperBASIC können rekursiv benutzt werden. Das heißt, eine Funktion oder Prozedur kann sich selbst direkt oder indirekt aufrufen.

Befehl	Funktion
DEFine FuNction	Definiert eine Funktion
DEFine PROCedure	Definiert eine Prozedur
RETurn	Verläßt eine Funktion oder Prozedur (gibt Daten aus einer Funktion zurück)
LOCal	Definiert lokale Daten in einer Funktion oder Prozedur

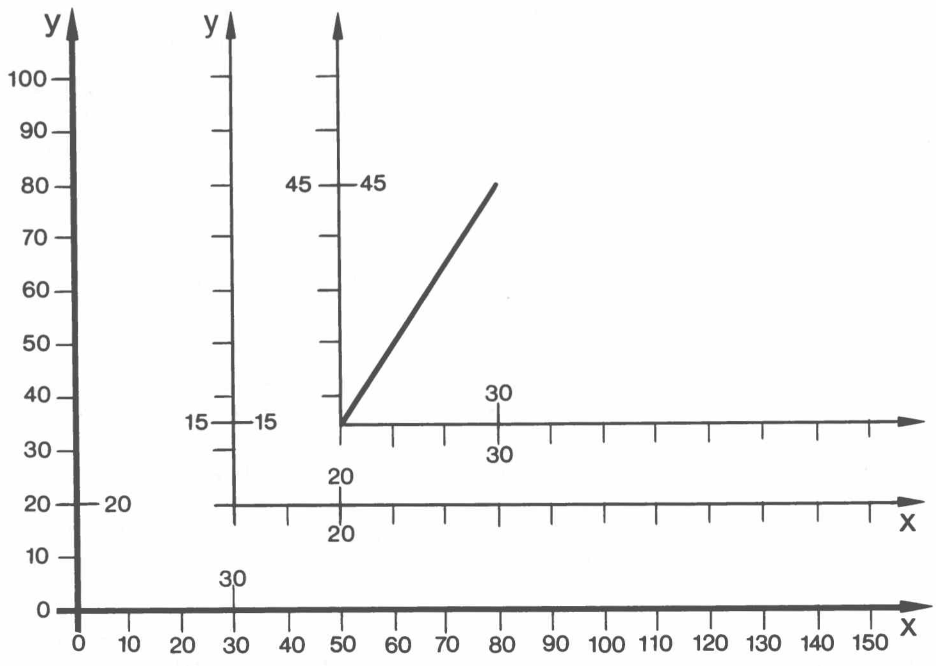
GRAFIK

Die Grafikprozeduren des QL erlauben, auf sehr einfache Art Grafiken aus Linien, Kreisen und Ellipsen aufzubauen. Mit den Prozeduren der normalen Grafik können Punkte durch Linien oder Kreisbögen miteinander verbunden werden. Die Prozeduren der **Turtle Grafik** ermöglichen es, von einem Punkt aus Linien mit einer bestimmten Länge und in einer bestimmten Richtung zu zeichnen. So einfach diese Prozeduren auch sein mögen, es lassen sich mit ihnen im Rahmen der Genauigkeit, die durch die Bildschirmauflösung gegeben ist, alle erforderlichen Grafiken darstellen – sei es zum Zwecke der ästhetischen Gestaltung, sei es zur technisch konstruktiven Arbeit.

Alle Grafikprozeduren arbeiten im **grafischen Koordinatensystem**. Die Form der Grafiken ist deshalb unabhängig von der Bildschirmbetriebsart. Die *Fenstertechnik* des QL und seine Möglichkeit, Maßstab und Lage des Koordinatensystems den jeweiligen Bedürnissen freizügig anzupassen, schaffen vielseitige Gestaltungsmöglichkeiten. So kann zum Beispiel durch Vergrößerung oder Verkleinerung des Ausgabefensters eine bestehende Grafik nach Belieben verkleinert oder vergrößert und an der gewünschten Stelle in der Gesamtdarstellung plaziert werden. (Einzelheiten dazu beim Begriff *Koordinaten.*)

Die Grafikprozeduren arbeiten im Prinzip auf einer beliebig großen Zeichenfläche. Durch das *Fenster*, an das die Ausgabe gerichtet ist, können Sie nur einen Teil der Zeichenfläche betrachten. Mit dem **SCALE**-Befehl können Sie den **Koordinatenursprung** verlegen und so das Fenster auf jeden beliebigen Ausschnitt der Zeichenfläche richten. (Siehe Begriff *Koordinaten.*)

Ein *Grafik-Cursor* wird von jeder Grafik-Prozedur auf den Punkt im Grafik-Koordinatensystem gesetzt, der zuletzt gezeichnet wurde. Nach dem Befehl **CIRCLE** oder **ELLIPSE** ist die Position des *Grafik-Cursors* unbestimmt.



Koordinaten bezogen auf den Grafik-Cursor

Die Grafikprozeduren können auf zwei verschiedene Arten benutzt werden. Einmal beziehen sie sich auf das mit **SCALE** oder durch den Standardwert festgelegte grafische Koordinatensystem. Mit dem Zusatz **_R** Schlüsselwort (z.B. **LINE_R**) beziehen sich die Koordinatenangaben auf die letzte Position des *Grafik-Cursors.* Dies ist so zu verstehen, als wäre für diesen Befehl der *Ursprung* des Grafik-Koordinatensystems an die letzte Position des *Grafik-Cursors* verlegt. Dabei ist zu beachten, daß z.B. mit

POINT 30,20

der Grafik-Cursor auf 30,20 gesetzt wird.

LINE_R 20,15 TO 30,45

hat dann folgende Wirkung: Die Koordinaten 20,15 beziehen sich auf den *Grafik-Cursor*, d. h. auf ein Koordinatensystem, dessen Ursprung jetzt in 30,20 liegt. Bei der Ausführung des Befehls wird der *Grafik-Cursor* sofort auf 20,15 gesetzt. Die Koordinaten 30,45 beziehen sich deshalb auf die neue Position des *Grafik-Cursors*, d. h. auf ein Koordinatensystem, dessen Ursprung in 20,15 liegt.

Befehl	Wirkung	
ARC	Zeichnet einen Kreisbogen	bezogen auf das grafische Koordinatensystem
CIRCLE	Zeichnet einen Kreis oder eine Ellipse	
LINE	Zeichnet eine Linie	
POINT	Zeichnet einen Punkt	
ARC_R	Zeichnet einen Kreisbogen	bezogen auf den Grafik-Cursor
CIRCLE_R	Zeichnet einen Kreis oder eine Ellipse	
LINE_R	Zeichnet eine Linie	
POINT_R	Zeichnet einen Punkt	
CURSOR	Positionieren von Text (mit 4 Parametern)	
FILL	Füllen von Figuren	
SCALE	Setzt den Maßstabsfaktor und den Ursprung des grafischen Koordinatensystems	

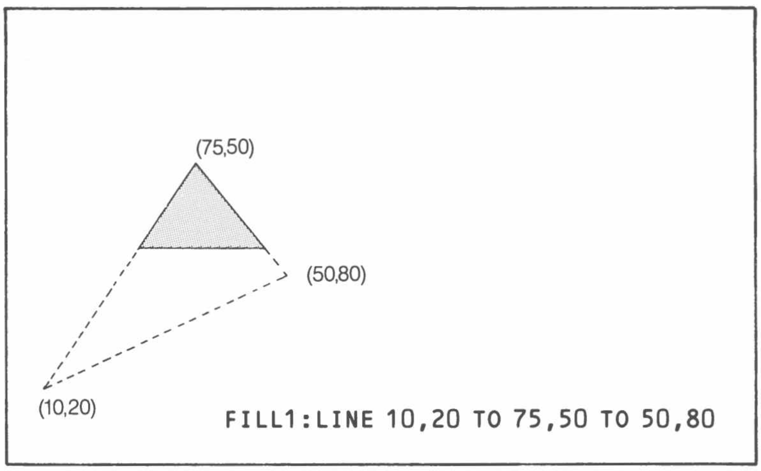
Füllen von Figuren

Figuren können auch mit einer Farbe ausgfüllt werden. Die Füllung erfolgt in der für das Fenster gültigen Schriftfarbe.

Der **FILL**-Algorithmus speichert eine Liste von Punkten, bevor die Figur gezeichnet wird. Sobald sich zwei Punkte auf einer waagerechten Linie befinden, kann die Figur dort gefüllt werden. Diese zwei Punkte werden durch eine Linie in der Schriftfarbe miteinander verbunden. Sobald sich erneut zwei Punkte auf gleicher waagerechter Linie befinden, wird der Vorgang wiederholt.

Der Befehl **FILL** muß immer wieder neu eingegeben werden, bevor eine neue, zu füllende Figur gezeichnet wird. Dadurch wird der Puffer gelöscht, indem die Liste von Punkten gespeichert wird.

Wirkungsweise des **FILL**-Algorithmus



Hinweis

FILL arbeitet nur bei konvexen Figuren korrekt. Andere Figuren müssen in konvexe Teilfiguren zerlegt werden. Jede Teilfigur ist dann für sich zu füllen.

JOYSTICK

Mit den Joystick-Anschlüssen **CTL1** und **CTL2** können zwei Joysticks an den QL angeschlossen werden.

Die Joysticks simulieren bestimmte Tastenbetätigungen, wenn sie auf eine bestimmte Art und Weise bewegt werden. Jedes Programm, das einen Joystick benutzt, muß diese Tastenbetätigungen interpretieren können. Die QL-Tastatur kann direkt mit dem **KEYROW**-Befehl gelesen werden.

	CTL1	CTL2
Bestätigung	**Taste**	**Taste**
Nach oben	Cursor-Taste nach oben	F4
Nach unten	Cursor-Taste nach unten	F2
Links	Cursor-Taste nach links	F1
Rechts	Cursor-Taste nach rechts	F3
Auslösen	Leertaste	F5

Kommentar

Über die Joystick-Anschlüsse können auch andere Steuereinheiten mit universellerem Einsatzgebiet an den QL angeschlossen werden.

Bei Joysticks für andere Computer, die einen 9poligen "D"-Stecker benutzen, ist ein Adapter erforderlich, damit sie mit dem QL benutzt werden können.

KANÄLE

Ein *Kanal* stellt die Verbindung zu einer QL-*Einheit* zur Eingabe oder Ausgabe von Daten her. Bevor ein Kanal benutzt werden kann, muß er mit dem **OPEN**-Befehl geöffnet werden. Bestimmte Kanäle müssen stets geöffnet sein, und zwar die Standardkanäle, die eine einfache Kommunikation mit dem QL über die Tastatur und den Bildschirm ermöglichen. Wird ein Kanal nicht mehr benutzt, so kann er mit dem **CLOSE**-Befehl geschlossen werden.

Ein Kanal wird durch eine *Kanalnummer* angesprochen. Eine Kanalnummer ist ein numerischer Ausdruck, vor dem ein # steht. Zum Beispiel:

```
OPEN #5, ser1
```

Mit diesem Befehl wird die Einheit SER1 mit Kanal Nummer 5 verknüpft. Soll ein Kanal geschlossen werden, so muß nur die Kanalnummer angegeben werden. Zum Beispiel:

```
CLOSE #5
```

Wird ein Kanal zu einer bestimmten *Einheit* geöffnet, so kann es notwendig sein, zusätzliche Parameter anzugeben. Das Netzwerk zum Beispiel benötigt eine *Stationsnummer*. Diese zusätzliche Information wird an den **OPEN**-Befehl angehängt. (Siehe die Begriffe *Einheiten* und *Peripherie-Erweiterung*.)

Daten können an einen Kanal ausgegeben werden, indem ein **PRINT**-Befehl für diesen Kanal ausgeführt wird. Hier wird dieselbe Methode benutzt, mit der auch die Ausgabe auf dem QL-Bildschirm angezeigt wird. Wird **PRINT** ohne einen Parameter benutzt, so geht die Ausgabe an den Standardkanal #1. Zum Beispiel:

```
100 OPEN_NEW #5, mdv1_test_datei
110 PRINT #5, "Dieser Text ist in der Datei test_datei"
120 CLOSE #5
```

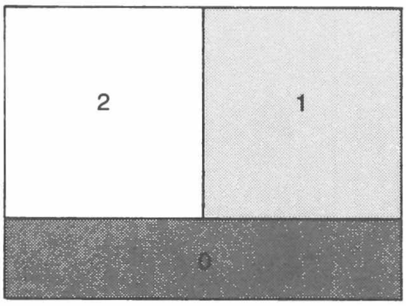
gibt den Text "Dieser Text ist in der Datei test-datei" an die Datei namens test-file aus. Die Datei muß unbedingt am Ende der Arbeit geschlossen werden, um zu gewährleisten, daß sämtliche Daten übertragen wurden.

Daten können mit **INPUT** aus einer Datei eingegeben werden. Daten können Zeichen für Zeichen mit **INKEY$** von einem Kanal eingegeben werden.

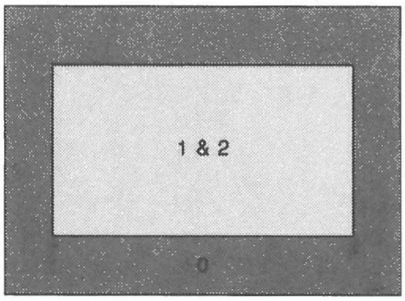
Ein Kanal kann als *Konsolkanal* geöffnet werden; die Ausgabe wird in ein bestimmtes *Fenster* auf dem QL-Bildschirm geleitet, während die Eingabe über die QL-Tastatur erfolgt. Sobald ein Konsolkanal geöffnet ist, sind Größe und Lage des Fensters festgelegt. Ist mehr als ein Konsolkanal aktiv, so kann mehr als ein Kanal gleichzeitig eine Eingabe anfordern. In diesem Fall kann der erforderliche Kanal durch Betätigung von **CTRL C** ausgewählt werden. Die wartenden Kanäle werden werden umgangen. Der Cursor in dem Fenster des ausgewählten Kanals blinkt auf.

Der QL verfügt über drei Standardkanäle, die beim Einschalten automatisch geöffnet werden. Jeder dieser Kanäle ist mit einem Fenster auf dem QL-Bildschirm verknüpft:

Kanal 0 – Befehls- und Fehlerkanal
Kanal 1 – Ausgabe- und Grafikkanal
Kanal 2 – Kanal für die Programmauflistung



Monitor



Fernsehgerät

Befehl	Funktion
OPEN	Öffnet einen Kanal für die Ein-/Ausgabe
CLOSE	Schließt einen vorher geöffneten Kanal
PRINT	Nimmt eine Ausgabe an einem Kanal vor
INPUT	Nimmt eine Eingabe von einem Kanal vor
INKEY$	Gibt ein Zeichen von einem Kanal ein

KOORDINATEN

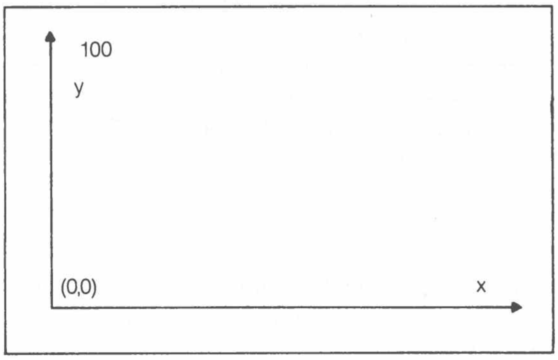
Grafik-Koordinaten-System

Im *Grafik-Koordinatensystem* arbeiten die Prozeduren der Grafik und der *Turtle Grafik*. Jedes Bildschirmfenster besitzt sein eigenes *Grafik-Koordinatensystem*. Es wird bei der Definition eines Fensters so festgelegt, daß sein *Ursprung* in der linken unteren Ecke des Fensters liegt. Die (senkrechte) y-Achse ist in 100 Längeneinheiten unterteilt. Die Längeneinheit auf der x-Achse ist immer gleich lang wie die Längeneinheit auf der y-Achse.

Mit dem SCALE-Befehl kann die Lage des *Koordinatenursprungs* sowie der Maßstabsfaktor verändert werden. Die Anweisung

SCALE m, x, y

bewirkt folgendes. Die y-Achse wird in m Längeneinheiten unterteilt. Der Ursprung des Koordinatensystems wird so festgelegt, daß die linke untere Ecke des Bildschirmfensters die Koordinaten x, y hat.



Grafik-Koordinatensystem

Geben Sie das folgende Programm ein und experimentieren Sie mit verschiedenen Maßstabsfaktoren und Fenstergrößen.

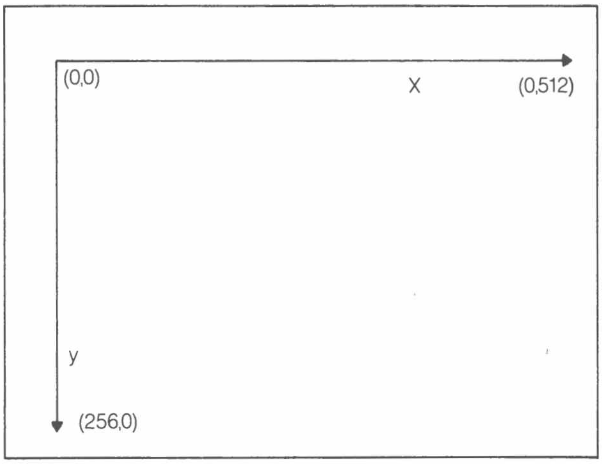
```
500 DEFine PROCedure ge
510 CIRCLE 0,0,10;-4,4,1;4,4,1
520 LINE 0,2 TO 0,-4
530 ARC -6,-4 TO 6,-4,PI/2
600 END DEFine
```

Pixel

Ein Pixel ist der kleinste Punkt, den der QL auf dem Bildschirm darstellen kann. Größe und Form eines Pixel hängen von der Betriebsart des Bildschirms ab. In der hochauflösenden Betriebsart können in der Waagerechten 512 Pixel, in der Senkrechten 256 Pixel dargestellt werden. In der niedrigauflösenden Betriebsart waagerecht und senkrecht jeweils 256 Pixel.

Pixel-Koordinaten-System

Das Maximalfenster des QL hat eine waagerechte Ausdehnung von 512 Pixeln und eine senkrechte Ausdehnung von 256 Pixeln. Innerhalb dieses Fensters liegt jedes Fenster, an das der QL Daten oder Grafiken ausgeben kann. Das *Pixel-Koordinatensystem* hat seinen *Ursprung* in der linken oberen Ecke eines Fensters. Die x-Achse ist beim Maixmalfenster in 512 Einheiten, die y- Achse in 256 Einheiten unterteilt, unabhängig, welche Bildschirmbetriebsart gewählt wird. Ein Heimfernseher kann allerdings das Maximalfenster nicht vollständig darstellen. An den Rändern geht einiges verloren. Das hat aber keinen Einfluß auf das Koordinatensystem.



Das Pixel-Koordinaten-System

Koordinatenangaben bei der Definition von Fenstern, z.B. beim **OPEN**- oder beim **WINDOW**-Befehl, beziehen sich immer auf das Maximalfenster.

Ausgabe am Bildschirmfenster

Jedes Bildschirmfenster hat sein eigenes *Pixel-Koordinatensystem.* Dessen *Ursprung* liegt immer in der linken oberen Ecke des Fensters. Die Koordinateneinheiten sind gleich wie im Maximalfenster. Die Länge der Koordinatenachsen wird bei der Definition oder Änderung des Fensters festgelegt.

Die Ausgabe von Schrift, z.B. mit **PRINT** oder **INPUT** und von Blöcken mit dem Befehl **BLOCK** benutzt das *Pixel-Koordinatensystem* des Fensters, an das die Ausgabe gerichtet ist.

Der Cursor legt die Position fest, an der die nächste Ausgab mit **PRINT** oder **INPUT** beginnt. Er kann mit den Befehlen **CURSOR** und **AT** gesetzt werden.

Im **CURSOR**-Befehl mit zwei Parametern werden die Parameter als *Pixel-Koordinaten* in dem angegebenen oder dem Standardfenster verstanden. Die Cursorposition bezeichnet die linke obere Ecke des Rechtecks, auf dem das nächste Zeichen ausgegeben wird.

Im **CURSOR**-Befehl mit vier Parametern wird das erste Parameterpaar als Koordinatenangabe im *grafischen Koordinatensystem* aufgefaßt. Das zweite Paar von Parametern wird aufgefaßt als Pixel-Koordinaten in einem Koordinatensystem, dessen *Ursprung* in dem Punkt liegt, der durch das erste Paar bezeichnet wird. In diesem Fall kann auch die Angabe negativer Pixel-Koordinaten sinnvoll sein.

Beispiel:

```
100 CSIZE 2,0: CLS
110 LINE 0,50 TO 40,85 TO 50,80 TO 85,100
120 CURSOR 40,85,-25,-10
130 PRINT 'UMSATZ'
```

Der **CURSOR**-Befehl mit vier Parametern eignet sich gut zur Beschriftung von Grafiken.

MATHE-MATISCHE FUNKTIONEN

SuperBASIC verfügt über die trigonometrischen und mathematischen Standardfunktionen.

Funktion	Name
COS	Kosnius
SIN	Sinus
TAN	Tangens
ATAN	Arcustangens
ACOT	Arcuskotangens
ACOS	Arcuskosinus
ASIN	Arcussinus
COT	Kotangens
EXP	Exponentialfunktion
LN	Natürlicher Logarithmus
LOG10	Zehnerlogarithmus
INT	Ganze Zahl
ABS	Absoluter Wert
RAD	Umsetzung in Radiant
DEG	Umsetzung in Grad
PI	Gibt den Wert von π zurück
RND	Liefert eine Zufallszahl
RANDOMISE	Legt eine neue Ausgangszufallszahl fest

MICRODRIVES

Microdrives stellen den hauptsächlichen permanenten Speicher bei dem QL dar. Jede Microdrive-Kassette verfügt über eine Kapazität von mindestens 100 KBytes. Der zur Verfügung stehende freie Speicherplatz wird von Qdos falls erforderlich in Form von Microdrive-Puffern zugewiesen, um die Leistung zu verbessern.

Jede leere Kassette muß vor der Benutzung *formatiert* werden und kann bis zu 255 Sektoren mit 512 Bytes pro Sektor umfassen. Qdos führt ein Inhaltsverzeichnis der auf der Kassette gespeicherten *Dateien.* Jede Microdrive-Datei wird mit einem SuperBASIC Standard-*Datei-* oder *Einheiten*-Namen gekennzeichnet.

Eine Kassette kann schreibgeschützt werden, indem der schmale Streifen von der Schreibschutzkerbe auf der rechten Seite abgenommen wird.

Neue leere QL Microdrive-Kassetten müssen mehrmals formatiert werden, um das Band für den Gebrauch vorzubereiten.

Allgemeine Pflege

Jede Microdrive-Kassette enthält 200 Zoll hochwertigen Videobandes, das mit einer Geschwindigkeit von 28 Zoll pro Sekunde abläuft. Pro Bandumlauf werden 7,5 Sekunden gerechnet.

Das Band darf **NIEMALS** mit den Fingern berührt werden. Außerdem darf nichts in die Kassette oder die Schutzhülle gesteckt werden.

Der Computer darf **NIEMALS** ein- oder ausgeschaltet werden, während die Kassetten eingelegt sind.

Die Kassetten müssen **IMMER** in den Schutzhüllen aufbewahrt werden, wenn sie nicht benutzt werden.

Die Kassetten müssen **STETS** langsam und vorsichtig in die Microdrives eingelegt und herausgenommen werden.

Vergewissern Sie sich **STETS**, daß die Kassette richtig eingelegt ist, bevor Sie einen Microdrive starten.

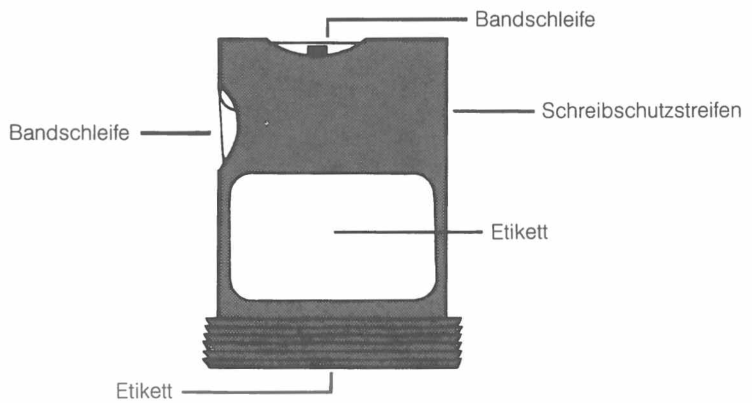
Der QL darf **NIEMALS** mit eingelegten Kassetten transportiert werden, selbst wenn er nicht in Betrieb ist.

Die Kassette darf **NIEMALS** berührt werden, während das Microdrive arbeitet.

Die Kassette **DARF NICHT** wiederholt eingelegt und herausgenommen werden, ohne daß der Microdrive gestartet wird.

Bandschleifen

Wird eine Bandschleife an einer der beiden in Abbildung 1 dargestellten Stellen sichtbar, so wird sie vorsichtig wieder in die Kassette zurückgeschoben. Hierzu wird ein faserfreier Gegenstand benutzt, z. B. der Schaft eines Bleistifts oder Kugelschreibers. Das Band darf weder hierbei noch sonst mit den Fingern berührt werden.



Microdrive-Kassette

Befehl	Funktion
FORMAT	Bereitet eine neue Kassette zur Benutzung vor
DELETE	Löscht eine Datei von einer Kassette
DIR	Listet die Dateien einer Kassette auf
SAVE SBYTES SEXEC	Sichert Daten auf einer Kassette
LOAD LBYTES EXEC MERGE	Lädt Daten von einer Kassette
OPEN_IN OPEN_NEW OPEN CLOSE	Öffnet und schließt Dateien
PRINT INPUT INKEY$	SuperBASIC Datei-E/A

Hinweis Wird versucht, eine schreibgeschützte Kassette zu beschreiben, so versucht der QL wiederholt, die Daten zu schreiben, gibt jedoch schließlich auf mit der Fehlermeldung "FEHLERHAFTER DATENTRAEGER".

Ein Monitor kann über den RGB-Anschluß auf der Rückseite des Computers an den QL angeschlossen werden. Der Anschluß erfolgt über einen 8poligen DIN-Stecker mit Kabel bei Farb-Monitoren oder über einen 3poligen DIN-Stecker mit Kabel bei Monochrom-Monitoren. Die Belegung der RBG-Anschlußstifte ist wie in der folgenden Tabelle dargestellt. Die Spalte, in der die Drahtfarbe angegeben wird, bezieht sich auf die Farbcodierung, die bei dem 8poligen Kabel und dem Stecker von Sinclair Research Limited benutzt wird. Die Bezeichnung der Steckerstifte entspricht dem nachfolgenden Diagramm.

Stecker-Stift	Funktion engl. deutsch	Signal		Isolationsfarbe bei dem QL RGB-Farbkabel
1	PAL / +12V	PAL-Videosignal / +12V	(4)	Orange
2	GND	Erde		Grün
3	VIDEO	Monochrom-Videosignal	(3)	Braun
4	CSYNC	Video-Synchronsignal	(2)	Gelb
5	VSYNC	Vertikal-Synchronsignal	(1)	Blau
6	GREEN	Grün	(1)	Rot
7	RED	Rot	(1)	Weiß
8	BLUE	Blau	(1)	Purpurrot

Ein Monochrom-Monitor kann mit einem abgeschirmten Kabel, das mit einem 3poligen oder 8poligen DIN-Stecker versehen ist, an dem QL angeschlossen werden. Nur die Steckerstifte 2 (Erde) und 3 (Videosignal) müssen über das Kabel an dem Monitor angeschlossen werden. Der Anschluß am Monitor variiert je nach benutztem Monitor, im allgemeinen wird jedoch ein Phono-Stecker benutzt. Der Monitor muß über einen 75-Ohm-Anschluß mit einem nicht invertierenden Videosignal von 1 V_{ss}verfügen (Industriestandard). Sowohl die 3poligen DIN-Stecker als auch die Phono-Stecker sind im Handel erhältlich.

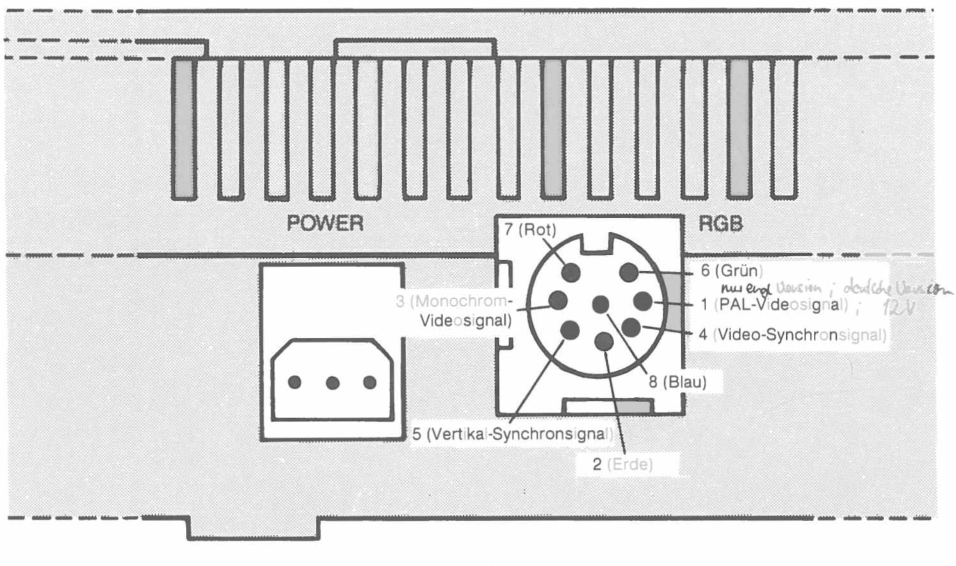


Diagramm des Monitor-Anschlusses von der Rückseite des QL gesehen, mit Angabe der Stiftnummer und Funktionen.

Ein RGB- (Farb-)Monitor kann mit einem Kabel und einem 8poligen DIN-Stecker für die QL-Seite angeschlossen werden. Der Anschluß auf der Monitorseite hängt von dem Monitor ab. (Hier gibt es keinen Industriestandard.) Häufig wird der Anschluß mit dem Monitor geliefert.

Ein PAL-Videomonitor oder der Videoeingang einiger Videorecorder **kann** unter Umständen mit dem QL benutzt werden. Nur die Steckerstifte 2 (Erde) und 1 (PAL-Video) müssen über ein Kabel an den Monitor oder Videorecorder angeschlossen werden.

NAME

Ein SuperBASIC-Name besteht aus einer Folge von Buchstaben, Zahlen und Unterstreichungszeichen.

Definition

Buchstabe:= | a..z
| A..Z

Zahl:= |1|2|3|4|5|6|7|8|9|0|

Name:= *Buchstabe* *[| *Buchstabe* | *Zahl* |__|]*

Beispiel:

a) `a`
b) `grenze_1`
c) `anfangs_wert`
d) `total`

Ein Name muß mit einem Buchstaben beginnen, auf den eine Folge von Buchstaben, Zahlen und Unterstreichungszeichen folgen kann. Ein Name kann eine Länge von bis zu 255 Zeichen aufweisen. Groß- und Kleinbuchstaben sind gleichbedeutend.

In dem SuperBASIC-System werden Namen zur Bezeichnung von *Variablen, Prozeduren, Funktionen, Wiederholschleifen* usw. benutzt.

Hinweis

Einem Namen kann KEINE andere Bedeutung zugewiesen werden, als die Bezeichnung von Programmteilen gegenüber SuperBASIC. SuperBASIC kann aus dem Namen nicht auf den Anwendungszweck schließen!

NETZWERK

Der QL kann mit bis zu 63 anderen QL-Computern verbunden werden. Sind mehr als zwei Computer in dem Netzwerk vorhanden, so muß jedem Computer (jeder Station) eine eindeutige *Stationsnummer* zugewiesen werden. Bei dem QL kann dies über den **NET**-Befehl erfolgen.

Die Daten werden in Form von Blöcken über das Netzwerk gesendet. Bei der normalen Datenübertragung zwischen zwei Stationen muß die empfangende Station den richtigen Empfang des Blocks bestätigen. Ist ein Block fehlerhaft, so fordert die empfangende Station eine erneute Übertragung an.

Die Benutzung einer Stationsnummer Null in dem Netzwerk hat eine besondere Bedeutung. Das Senden an **neto_0** wird als Rundsendung bezeichnet: jede auf diese Art und Weise gesendete Meldung kann von jeder Station gelesen werden, die auf **neti_0** hört. Hier wird darauf hingewiesen, daß die normale Überprüfung des richtigen Empfangs einer Meldung bei Rundsendungen nicht benutzt wird. So ist die Verbreitung von Nachrichten mit einer Länge von mehr als einem Block (255 Bytes) nicht verläßlich.

Eine Station im Netzwerk, die auf die eigene Stationsnummer hört (z. B. **NET3:LOAD neti_3**), kann Daten von jeder Station empfangen, die an sie sendet.

Befehl	Funktion
NET	Weist eine Stationsnummer innerhalb des Netzwerkes zu
OPEN CLOSE	Öffnet einen Netzwerk-Kanal Schließt einen Netzwerk-Kanal
PRINT INPUT INKEY$	Netzwerk-E/A
LOAD SAVE LBYTES SBYTES EXEC SEXEC LRUN MRUN MERGE	Laden und Sichern über das Netzwerk

Kommentar

Sollen mehrere QL-Computer in dem Netzwerk miteinander verbunden oder soll ein langes Kabel benutzt werden, so muß die Verbindung mit einem kapazitätsarmen, zweiadrigen Kabel vorgenommen werden, wie beispielsweise einem 3 A Kabel oder einem Klingeldraht. Hier muß darauf geachtet werden, daß die Mittelkontakte des Klinkensteckers und die Außenseiten miteinander verbunden werden. Auch wenn die Software insgesamt 63 Stationen bearbeiten kann, werden Sie feststellen, daß die Hardware je nach Typ des Kabels nicht mehr als 100 m ansteuern kann.

Werden nur wenige Computer mit den gelieferten Kabeln miteinander verbunden, so brauchen Sie sich hier keine Gedanken zu machen.

OPERATOREN

Priorität

Die Priorität der SuperBASIC-Operatoren wird in der Tabelle unten angegeben. Bei Operationen gleicher Priorität werden die entsprechenden Operationen von links nach rechts ausgeführt. Die festgelegte Priorität der SuperBASiC-Operatoren kann verändert werden, indem die entsprechenden Teile des Ausdrucks in Klammern gesetzt werden.

Höchste Priorität:	Unäres Plus und Minus
	String-Verkettung
	INSTR
	Potenzierung
	Multiplikation, Division, MOD und ganzzahlige Division
	Addition und Subtraktion
	Logischer Vergleich
	NICHT (bitweise oder logisch)
	UND (bitweise oder logisch)
Niedrigste Priorität:	ODER und EXCLUSIV ODER (bitweise oder logisch)

Operator	Typ	Funktion
=	Gleitkommazahl, String	Logischer Typ 2 Vergleich
= =	Numerisch, String	fast gleich** (Typ 3 Vergleich)
+	Numerisch	Addition
–	Numerisch	Subtraktion
/	Numerisch	Division
*	Numerisch	Multiplikation
<	Numerisch, String	kleiner als (Typ 2 Vergleich)
>	Numerisch, String	größer als (Typ 2 Vergleich)
< =	Numerisch, String	kleiner als oder gleich (Typ 2 Vergleich)
> =	Numerisch, String	größer als oder gleich (Typ 2 Vergleich)
< >	Numerisch, String	ungleich (Typ 3 Vergleich)
&	String	Verknüpfung
& &	Bitweise	AND
||	Bitweise	OR
^ ^	Bitweise	XOR
~	Bitweise	NOT
OR	Logisch	OR
AND	Logisch	AND
XOR	Logisch	XOR
NOT	Logisch	NOT
MOD	Ganze Zahl	Modul
DIV	Ganze Zahl	Division
INSTR	String	Typ 1 String Vergleich
^	Gleitkommazahl	Potenzieren
–	Gleitkommazahl	unäres Minus
+	Gleitkommazahl	unäres Plus

** fast gleich – gleich zu 1 Teil in 10^7

Wenn die spezifizierte logische Operation wahr ist, wird ein Wert ungleich Null zurückgegeben. Ist die Operation nicht wahr, wird ein Wert gleich Null zurückgegeben.

Hinweis Weitere Informationen sind unter dem Begriff *String-Vergleich* zu finden.

PERIPHERIE-ERWEITERUNG

Über den Erweiterungs-Anschluß können zusätzliche Peripheriegeräte an dem QL angeschlossen werden. Über den Anschluß stehen folgende Signale zur Verfügung:

Signal		Pin		Signal
GND	a	1	b	GND
D3	a	2	b	D2
D4	a	3	b	D1
D5	a	4	b	D0
D6	a	5	b	ASL
D7	a	6	b	DSL
A19	a	7	b	RDWL
A18	a	8	b	DTACKL
A17	a	9	b	BGL
A16	a	10	b	BRL
CLKCPU	a	11	b	A15
RED	a	12	b	RESETCPUL
A14	a	13	b	CSYNCL
A13	a	14	b	E
A12	a	15	b	VSYNCH
A11	a	16	b	VPAL
A10	a	17	b	GREEN
A9	a	18	b	BLUE
A8	a	19	b	FC2
A7	a	20	b	FC1
A6	a	21	b	FC0
A5	a	22	b	A0
A4	a	23	b	ROMOEH
A3	a	24	b	A1
DBGL	a	25	b	A2
SP2	a	26	b	SP3
DSMCL	a	27	b	IPL0L
SP1	a	28	b	BERRL
SP0	a	29	b	IPL1L
VP12	a	30	b	EXTINTL
VM12	a	31	b	VIN
VIN	a	32	b	VIN

Bei dem Anschluß an dem QL handelt es sich um einen 64poligen indirekten Steckverbinder nach DIN 416112.

Ist ein 'L' an den Signalnamen angehängt, so bedeutet dies, daß das Signal im Logikpegel Low aktiv ist.

Ausgangssignale der QL-Peripherieeinheit

Signal	Funktion
A0..A19	68008-Adreßleitungen
RDWL	Lesen/Schreiben
ASL	Adreß-Freigabesignal
DSL	Daten-Freigabesignal
BGL	Bus-Freigabesignal
DSMCL	Daten-Freigabesignal Master Chip
CLKCPU	Taktgeber der Zentraleinheit
E	68008 Peripheriegeräte-Taktgeber
RED	Rot
BLUE	Blau
GREEN	Grün
CSYNCL	Video-Synchronsignal
VSYNCH	Vertikal-Synchronsignal
ROMOEH	Freigabesignal für ROM-Ausgang
FC0	Prozessorstatus
FC1	Prozessorstatus
FC2	Prozessorstatus
RESETCPUL	Rücksetzung der Zentraleinheit

Eingangssignale der QL-Peripherieeinheit

Signal	Funktion
DTACKL	Datenrückmeldung
BRL	Bus-Anforderung
VPAL	Gültige Peripheriegeräte-Adresse
IPL0L	Unterbrechungsebene 5
IPL1L	Unterbrechungsebene 2
BERRL	Busfehler
EXTINTL	Externe Unterbrechung
DBGL	Datenbusübernahme

Bidirektionale Signale der QL-Peripherieeinheit

Signal	Funktion
D0 . . . D7	Datenleitungen

Verschiedene

Signal	Funktion
SP0 . . . SP3	Auswahl von Peripheriegerät 0 bis 3
VIN	9 V Gleichspannung – 500 mA maximal
VM12	– 12 V
VP12	+ 12 V
GND	Erde

Die folgende Beschreibung der Erweiterung durch QL-Peripheriegeräte reicht für die tatsächliche Implementierung einer Erweiterungseinheit nicht aus. Sie soll nur einen allgemeinen Überblick über das Erweiterungsverfahren vermitteln.

Maximal 16 Peripheriegeräte können zu dem QL hinzugefügt werden. Ein einzelnes Peripheriegerät kann direkt an dem Erweiterungsanschluß des QL angeschlossen werden, während mehrere Peripheriegeräte an dem Erweiterungsmodul des QL angeschlossen werden müssen, das wiederum über eine Puffer-Platine an dem QL-Erweiterungsanschluß angeschlossen wird.

In diesem Zusammenhang umfaßt der Ausdruck "Einheit" auch eine Speichererweiterung. Auch wenn sich die für die Speichererweiterung vorgesehenen Bereiche in der QL-Speichereinteilung von den Bereichen für die Erweiterungseinheiten unterscheiden, wird doch dasselbe Prinzip benutzt. Es kann jeweils nur eine Erweiterungseinheit an dem QL angeschlossen werden. Der Adreßbereich für die Peripheriegeräte-Erweiterungen in der Speichereinteilung des QL läßt 16 kBytes pro Peripheriegerät zu. Dieser Bereich muß den E/A-Speicher für den Treiber und den Code für den Treiber selbst umfassen.

Qdos umfaßt Einrichtungen für die Warteschlangenverwaltung und eine einfache serielle E/A, die beim Schreiben von Einheitentreibern von Nutzen sein kann.

Die Position jedes Peripheriegerätes in der gesamten Speichereinteilung des QL wird durch die Leitung zur Auswahl der Peripheriegeräte bestimmt: SP0, SP1, SP2 und SP3. Diese Auswahlleitungen liefern ein Signal, das der Anschlußposition in dem QL-Erweiterungsmodul entspricht. Für eine auszuwählende Einheit muß der Adresseneingang von den Adreßleitungen A14, A15, A16 und A17 den Signalen von SP0, SP1, SP2 bzw. SP3 entsprechen.

PROGRAMM

Ein SuperBASIC-Programm besteht aus einer Folge von SuperBASIC-*Anweisungen*, wobei vor jeder Anweisung eine *Zeilennummer* steht. Die Zeilennummern liegen in dem Bereich von 1 bis 32767.

Befehl	Funktion
RUN	Startet ein geladenes Programm
LRUN	Lädt ein Programm von einer Einheit und startet es
CTRL LEERTASTE	Stoppt ein Programm

Syntax: *Zeilennummer:=* *[*Ziffer*]* {Bereich 1..32767}
*[*Zeilennummer Anweisung*[:*Anweisung*]*]*

Beispiel:

a)
```
100 PRINT ''Das ist eine gültige Zeilennummer''
RUN
```

b)
```
100 REMark Ein kleines Programm
110 FOR vordergrund = 0 TO 7
120   FOR kontrast = 0 TO 7
130     FOR punktmuster = 0 TO 3
140       PAPER vordergrund, kontrast,
          punktmuster
150       CURSOR 0,70
160       FOR n = 0 TO 2
170         SCROLL 2,1
180         SCROLL -2,2
190       END FOR n
200     END FOR punktmuster
210   END FOR kontrast
220 END FOR vordergrund
RUN
```

QDOS

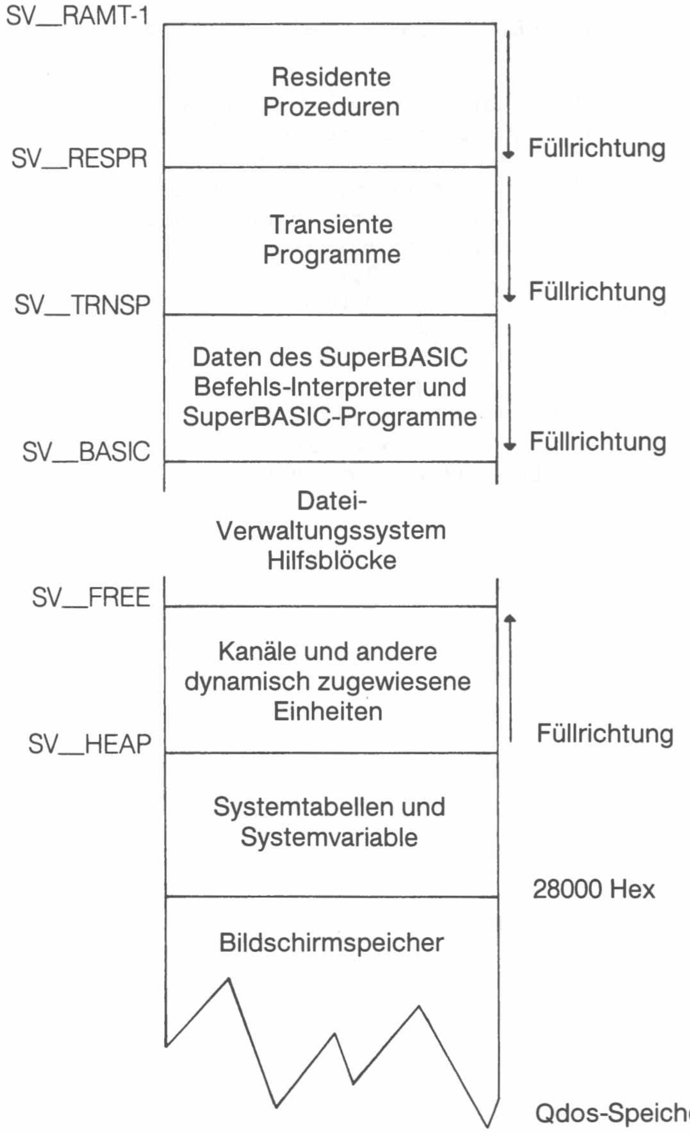
Qdos ist das QL-Betriebssytem, das folgende Funktionen überwacht:

- Abarbeitung von Programmbereichen und Zuweisung der Betriebsmittel
- Bildschirm-E/A (einschließlich Fenster-Bearbeitung)
- Microdrive-E/A
- Datenübertragung über Netzwerke und serielle Kanäle
- Tastatureingabe
- Speicherverwaltung.

Speichereinteilung

Eine vollständige Beschreibung von Qdos würde den Rahmen dieses Handbuches sprengen. Deshalb wird hier nur eine kurze Beschreibung vorgenommen.

Die Organisation des System-RAM wird von dem Qdos Betriebssystem bestimmt und ist folgendermaßen definiert:



Qdos-Speichereinteilung

Mit den Begriffen SV_RAMT, SV_RESPR, SV_TRNSP, SV_BASIC, SV_FREE und SV_HEAP werden die Adressen innerhalb des QL dargestellt. Diese Begriffe werden von SuperBASIC oder dem Qdos-Betriebssystem nicht erkannt. Außerdem können die dargestellten Adressen sich ändern, während das System läuft.

sv_ramt — **RAM-Top**
Er variiert je nach den Speichererweiterungsplatinen, die an das System angeschlossen sind.

sv_respr — **Residente Prozeduren**
Die residenten Prozeduren werden in den Anfang des RAM geladen. Mit der **RESPR**-Funktion kann Platz in dem Bereich der residenten Prozeduren zugwiesen werden. Dieser Platz kann jedoch

nur durch Rücksetzung des QL wieder freigegeben werden. Im Maschinencode geschriebene residente Prozeduren können zu der SuperBASIC-Namensliste hinzugefügt werden und werden so zu Erweiterungen des SuperBASIC-Systems.

sv_trnsp **Transiente Programme**
Transiente Programme werden direkt unter den residenten Prozeduren geladen. Jedes Programm muß selbständig sein, d.h. muß Platz für die eigenen Daten und den eigenen Stack aufweisen. Es muß positions-unabhängig sein oder von einem speziell geschriebenen Link-Ladeprogramm geladen werden. Ein transientes Programm wird mit dem **EXEC**-Befehl aus BASIC ausgeführt. Soll es aus Qdos ausgeführt werden, so muß es als *Job* aktiviert werden.
Der transiente Programmbereich kann auch für die Speicherung von Daten benutzt werden. Diese Daten werden jedoch von Qdos als Job behandelt und dürfen demzufolge nicht aktiviert werden.

sv_basic **SuperBASIC-Bereich**
Dieser Bereich enthält sämtliche geladenen SuperBASIC-Programme und zugehörigen Daten. Dieser Bereich wird erweitert und verkürzt, wobei freier Platz nach Bedarf benutzt wird.

sv_free **Freier Platz**
In dem freien Platz erstellt das Qdos-Dateiverwaltungssystem untergeordnete Microdrive-Blöcke, d.h. Kopien von Microdrive-Blöcken, die in dem RAM gespeichert werden.

sv_heap **Bereiche für dynamisch zugewiesene Systemdaten**
Dieser Bereich wird vom System zur Speicherung der Definition von Datenkanälen usw. benutzt und stellt den Arbeitsspeicher für das E/A-Subsystem dar. Transiente Programme können sich in diesem Bereich über Qdos-Systemaufrufe Arbeitsplatz zuweisen.

Systemtabellen/Systemvariable
Dieser Bereich befindet sich direkt über dem Bildschirmspeicher. Die Systemtabellen und der Supervisor-Stack sind über den Systemvariablen abgelegt.

Systemaufrufe

Systemaufrufe werden von Qdos im **Supervisor-Modus** verarbeitet. Im Supervisor-Modus läßt Qdos nicht zu, daß ein anderer Job den Prozessor benutzt. Auf diese Weise verarbeitete Systemaufrufe werden als **atomar** bezeichnet, d.h. der Systemaufruf führt die Verarbeitung bis zum Ende aus, bevor der Prozessor freigegeben wird. Einige Systemaufrufe sind nur **teilweise atomar,** d.h. sobald ihre Primärfunktion ausgeführt wurde, geben sie den Prozessor ggf. frei. Werden nicht ausdrücklich andere Anforderungen gestellt, so sind sämtliche E/A-Systemaufrufe teilweise atomar.

Standardmäßig wird ein Systemaufruf ausgeführt, indem eine Verzweigung zu einem der Qdos-Systemvektoren mit entsprechenden Parametern in den Prozessor-Registern vorgenommen wird. Die von Qdos im Anschluß an einen Systemaufruf ausgeführten Schritte hängen von dem jeweiligen Aufruf und dem Gesamtstatus des Systems zum Zeitpunkt des Aufrufs ab.

Ein-/Ausgabe

Qdos unterstützt *Multitasking.* Demzufolge könnten mehrere Operationen gleichzeitig auf eine Datei zugreifen. Das Qdos-Dateiverwaltungssystem kann Dateien verarbeiten, die als **exklusive** Dateien oder als **gemeinsam** benutzte Dateien geöffnet wurden. In eine gemeinsam benutzte Datei kann nicht geschrieben werden. QL-Einheiten werden von dem **seriellen E/A-Subsystem** verarbeitet. Das Dateiverwaltungssystem und das serielle E/A-Subsystem bilden gemeinsam das **umleitbare E/A-System.** Wie sich schon aus dem Namen ergibt, kann jede Datenausgabe über dieses System zu einer anderen Einheit umgeleitet werden, die ebenfalls von dem umleitbaren E/A-System unterstützt wird.

Von Qdos werden dieselben Einheitennamen benötigt wie von SuperBASIC. Sie werden in diesem Abschnitt unter *Einheiten* erläutert. Die Gruppe der mit dem QL gelieferten Standardeinheiten kann erweitert werden.

Einheiten

Die in dem System enthaltenen Standardeinheiten werden in diesem Kapitel unter **Einheiten** erläutert. Weitere Einheiten können zu dem System hinzugefügt werden. Nachdem ihnen ein Name zugewiesen wurde (z. B. SER1, NET), kann auf sie wie auf andere QL-Einheiten zugegriffen werden.

Multitasking

Jobs können die Zentraleinheit je nach Priorität und Konkurrenz mit anderen Jobs in dem System gemeinsam benutzen. Die unter Qdos ausgeführten Jobs können einen von drei Zuständen aufweisen:

Aktiv: Sie können die Betriebsmittel des Systems gemeinsam benutzen. Ein Job in diesem Status läuft unter Umständen nicht ständig, sondern kann die Zentraleinheit je nach Priorität benutzen.

Suspendiert: Der Job kann ausgeführt werden, wartet jedoch auf einen anderen Job oder eine E/A. Ein Job kann ständig oder während einer bestimmten Zeit suspendiert werden.

Inaktiv: Der Job kann nicht ausgeführt werden. Er hat eine Priorität von 0 und kann somit niemals die Zentraleinheit benutzen.

Qdos führt mit einer Häufigkeit, die mit der 50 Hz Bildwiederholungsfrequenz gekoppelt ist, automatisch eine Neuplanung des Systems durch. Ferner wird nach bestimmten Systemaufrufen eine Neuplanung des Systems durchgeführt.

Beispiel: Mit diesem Programm wird die Echtzeituhr als unabhängiger Job auf dem Bildschirm ausgegeben.

Zuerst wird dieses Programm mit einer formatierten Kassette in Microdrive 2 mit **RUN** ausgeführt. Dadurch wird ein Maschinencodeprogramm namens Uhr erzeugt. Nun muß gewartet werden, bis das Microdrive stoppt. Als nächstes wird die Uhr mit dem **SDATE**-Befehl eingestellt.

Danach wird:

```
EXEC mdv2_uhr
```

eingegeben. Nun wird in der oberen rechten Ecke des Befehlsfensters kontinuierlich die Uhrzeit angezeigt.

```
100 c=RESPR(100)
110 FOR i =0 TO 68 STEP 2
120 READ x: POKE_W i+c,w
130 END FOR i
140 SEXEC mdv2_uhr,c, 70,128
1000 DATA 29439,29697,28683,20033,17402
1010 DATA 48,13944,200,20115,12040
1020 DATA 28691,20033,17402,74,-27698
1030 DATA 13944,236,20115,8279,-11314
1040 DATA 13944,208,20115,16961,16962
1050 DATA 30463,28688,20035,24794
1060 DATA 0,7,240,10,272,200
```

Hinweis

Mit Zeile 1060 werden die Position und Farbe des Uhr-Fensters festgelegt
– folgende Datenelemente werden in dieser Reihenfolge angezeigt:
Randfarbe/Breite, Papier-/Schriftfarbe, Fensterbreite, Höhe,
X-Ursprung, Y-Ursprung

Hier handelt es sich um Byte-Paare, die mit **POKE_W** als Worte eingegeben werden.

Der X-Ursprung und der Y-Ursprung (das letzte Datenelement) muß 272 bzw. 202 im Monitor-Modus oder 240 bzw. 216 im TV-Modus betragen.

Das Wort für Papier- und Schriftfarbe wird beispielsweise als 256* paper + ink gebildet. Also entsprechen weißes Papier und rote Schriftfarbe 256* 7 + 2 = 1794.

ROM-KASSETTEN-ANSCHLUSS

Über ihn kann Software von einer Sinclair QL ROM-Kassette in dem QL-System benutzt werden. Die ROM-Kassette kann Software enthalten, mit der das Verhalten des SuperBASIC-Systems direkt geändert werden kann. Auf der Kassette kann folgendes stehen:

a) Die Software, die anstelle des SuperBASIC-Systems oder mit dem SuperBASIC-System benutzt werden soll. Zum Beispiel:

   Assembler
   Compiler
   Testprogramme
   Anwendungs-Software
   usw.

b) Software zur Erweiterung des SuperBASIC-Systems. Zum Beispiel:

   Sonderprozeduren
   usw.

ZX-ROM-Kassetten können bei dem QL nicht benutzt werden.

Anschlußbelegung

—	a	1	b	VDD
A12	a	2	b	A14
A7	a	3	b	A13
A6	a	4	b	A8
A5	a	5	b	A9
SLOT	a	6	b	SLOT
A4	a	7	b	A11
A3	a	8	b	ROMOEH
A2	a	9	b	A10
A1	a	10	b	A15
A0	a	11	b	D7
D0	a	12	b	D6
D1	a	13	b	D5
D2	a	14	b	D4
GND	a	15	b	D3

Seite b ist die Oberseite des Anschlusses und Seite a die Unterseite.

Signal	Funktion
A0 .. A15	Adreßleitungen
D0 .. D8	Datenleitungen
ROMOEH	Freigabesignal für die ROM-Ausgabe
VDD	5 V
GND	Erde

Hinweis

Eine ROM-Kassette darf niemals eingelegt oder herausgenommen werden, während der QL eingeschaltet ist.

SCHLEIFEN

Die Schleifen bei SuperBASIC werden von zwei grundlegenden Programm-Konstruktionen gesteuert. Jede Konstruktion muß gegenüber SuperBASIC beschrieben werden:

REPEAT *Name*	FOR *Name = Bereich*
Anweisungen	*Anweisungen*
END REPEAT *Name*	END FOR *Name*

Diese beiden Konstruktionen werden in Verbindung mit zwei anderen SuperBASIC-Anweisungen benutzt:

NEXT *Name*	EXIT *Name*

Durch Ausführung einer **NEXT**-Anweisung wird das Programm entweder mit der Anweisung fortgesetzt, die auf die entsprechende **FOR**- oder **REPeat**-Anweisung folgt, oder – wenn ein **FOR**-Bereich erschöpft ist – mit der Anweisung fortgesetzt, die auf die **NEXT**-Anweisung folgt.

Durch Ausführung einer **EXIT**-Anweisung wird das Programm mit der Anweisung fortgesetzt, die auf die **END FOR**- oder **END REPeat**-Anweisung folgt, die von der **EXIT**-Anweisung ausgewählt wurde. Mit **EXIT** können mehrere Ebenen von verschachtelten Schleifen-Strukturen beendet werden. **EXIT** muß stets in **REPeat**-Schleifen benutzt werden, um die Schleife bei einer bestimmten Bedingung zu beenden.

Bei einer Kombination von **NEXT, EXIT**- und **END**-Anweisungen kann zu **FOR**- und **REPeat**-Schleifen ein **Schleifennachsatz** hinzugefügt werden. Ein Schleifennachsatz besteht aus einer Folge von SuperBASIC-Anweisungen, die ausgeführt werden, wenn es innerhalb der Schleife zu einer bestimmten Bedingung kommt:

```
FOR Name = for_Liste
  Anweisungen ◄──┐       Beenden──┐
NEXT Name_Nächste─┘               │
  Nachsatz                        │
END FOR Name ◄────────────────────┘
```

Der Schleifennachsatz wird nur verarbeitet, wenn die **FOR**-Schleife normal beendet wird. Wird die Schleife über eine **EXIT**-Anweisung beendet, so wird die Verarbeitung bei der **END FOR**-Anweisung fortgesetzt, und der Nachsatz wird nicht verarbeitet.

In einer **REPeat**-Schleife ist eine ähnliche Konstruktion möglich:

```
REPeat Name ◄─────────────────────┐
  Anweisungen                     │
IF Bedingung THEN NEXT Name ──────┘
  Nachsatz
END REPeat Name
```

Diesmal wird der Einsprung in den Schleifennachsatz durch die **IF**-Anweisung gesteuert. Der Nachsatz wird je nach der Bedingung in der **IF**-Anweisung verarbeitet oder nicht. Der Einsprung in den Nachsatz kann auch mit einer **SELect**-Anweisung gesteuert werden.

SPEICHER-EINTEILUNG

Der QL enthält einen Motorola 68008 Mikroprozessor, der 1 MB Speicher adressieren kann, d. h. von 00000 Hex bis FFFFF Hex. Die Benutzung der Adressen innerhalb dieses Bereiches ist von Sinclair Research wie folgt definiert:

Adresse	Speicher	Verwendung
FFFFF – C0000	RESERVIERT	E/A-Erweiterung
C0000 – 40000	RESERVIERT	Zusatz-RAM
40000 – 28000	RAM 96K Bytes	Haupt-RAM
28000 – 20000	RAM 32K Bytes	Bildschirm-RAM
20000 – 18000	E/A	QL E/A
18000 – 0C000	ROM 16K Bytes	Steckbarer ROM
0C000 – 00000	ROM 48K Bytes	System-ROM

Speichereinteilung

Der Bildschirm-RAM ist als eine Folge von 16-Bit-Wörtern organisiert, die bei Adresse 20000 Hex beginnt und in der Reihenfolge der Rasterabtastung fortgesetzt wird, d. h. von links nach rechts auf jeder Bildschirmzeile und vom Anfang bis zum Ende des Bildes. Die Bits innerhalb jedes Wortes sind so organisiert, daß ein links stehendes Pixel immer signifikanter ist als ein rechts stehendes Pixel (d. h. das Pixel-Muster auf dem Bildschirm sieht wie das Bit-Muster aus). Die Farbinformation in den beiden Bildschirm-Betriebsarten erfolgt jedoch unterschiedlich:

Höherwertiges Byte AO = 0	Niederwertiges Byte AO = 1	Betriebsart
GGGGGGGG	RRRRRRRR	Betriebsart 512 (niedrige Auflösung)
GFGFGFGF	RBRBRBRB	Betriebsart 256 (hohe Auflösung)

G – Grün B – Blau R – Rot F – Blinken

Wird das Blink-Bit gesetzt, so wird der Blink-Status eingeschaltet und die Hintergrundfarbe für das Blinken auf den Wert festgeschrieben, der mit R, G und B für dieses Pixel angegeben wird.

Im hochauflösenden Modus wird die gleichzeitige Angabe von Rot und Grün von der Hardware als Weiß interpretiert.

Hinweis

Die Benutzung reservierter Bereiche in der Speichereinteilung kann zu einer Inkompatibilität mit künftigen Sinclair Produkten führen. Die Ausgabe an Adressen, die als E/A-Adressen für Peripheriegeräte definiert sind, kann zu einem nicht vorhersehbaren Verhalten führen. Es wird empfohlen, daß in diese Bereiche NICHT geschrieben wird, und daß sie nicht für andere Zwecke benutzt werden. Werden als Microdrive-Puffer benutzte Bereiche mit **POKE** geändert, so kann es zu einer Zerstörung der

Microdrive-Daten und zu einem Datenverlust kommen. Werden benutzte Bereiche, wie beispielsweise Systemtabellen, mit **POKE** geändert, so kann es zu einem Systemabsturz und zu einem Daten- und Programmverlust kommen.

Die gesamte E/A muß entweder mit den entsprechenden SuperBASIC-Befehlen oder den Routinen des *Qdos-Betriebssystems* ausgeführt werden.

STANDARD-WERTE

Viele SuperBASIC-Befehle benötigen eine bestimmte Anzahl von Parametern. Oft ist es aber so, daß für einige Parameter fast immer dieselben Werte eingesetzt werden. Es wäre mühsam, bei bestimmten Befehlen immer wieder dieselben Parameterwerte einzugeben. In solchen Fällen setzt SuperBASIC immer dann einen sinnvollen Standardwert für einen Parameter ein, wenn Sie nicht selbst einen anderen Wert eingeben.

Immer wenn in diesem Handbuch von Standardwerten die Rede ist, handelt es sich um solche Vorschläge, die von SuperBASIC automatisch eingesetzt werden. Bei den einzelnen Befehlen und Begriffen ist angegeben, welche Standardwerte vorgesehen sind.

Beispiel:

```
PRINT ''Ausgabe an . . .''
```

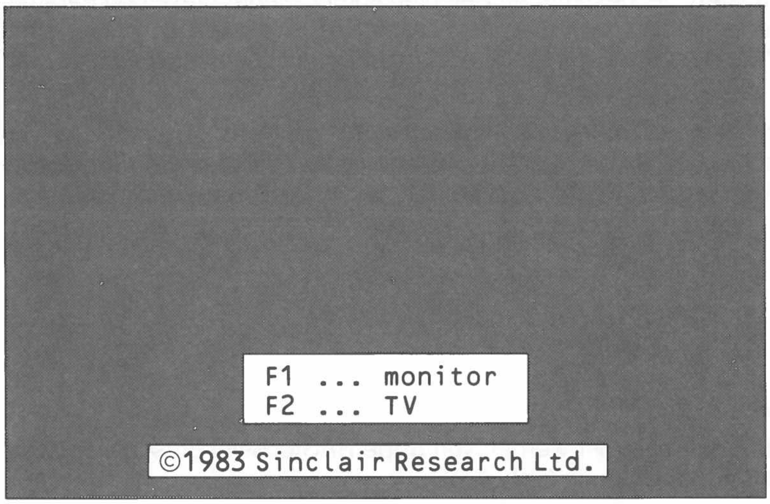
steht für

```
PRINT#1, ''Ausgabe an . . .''
```

Die Parameter für den Kanal #1 werden von SuperBASIC beim Einschalten automatisch eingesetzt. (Siehe Begriff *Einheiten*).

STARTEN

Unmittelbar nach dem Einschalten (oder Rücksetzen) führt der QL einen RAM-Test aus, bei dem ein willlkürliches Muster auf dem Bildschirm angezeigt wird. Nachdem der RAM-Test beendet ist, wird der Bildschirm gelöscht und der Copyright-Vermerk angezeigt.



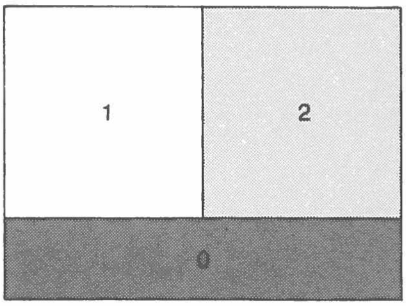
Nach dem Start zeigt der QL den Copyright-Vermerk an und fragt, ob er mit einem Fernsehgerät oder einem Monitor benutzt wird. Je nach Antwort legt der QL verschiedene Bildschirm-Betriebsarten und Fenstergrößen fest.

Wird ein Monitor benutzt, so betätigen Sie **F1**. Wird ein Fernsehgerät benutzt, so wird **F2** betätigt.

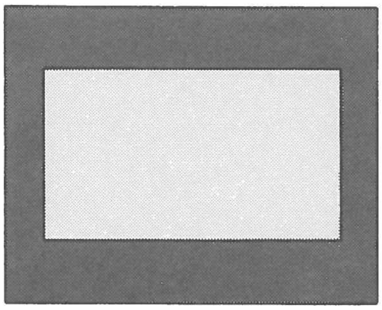
Der QL kann nach dem Start einen Selbstladevorgang mit Programmen durchführen, die entweder auf der ROM-Kassette oder auf einer Kassette in Microdrive 1 stehen. Enthält die ROM-Kassette ein Programm für den automatischen Start, so wird das Programm auf der ROM-Kassette ausgeführt. Wird dort kein Programm für den automatischen Start gefunden, so prüft der QL, ob in Microdrive 1 eine Kassette eingelegt ist. Wird eine Kassette gefunden und enthält sie eine Datei namens **BOOT**, so wird diese geladen und ausgeführt.

Standardbildschirm

Der QL verfügt über drei Standardkanäle, die mit drei Standardfenstern verknüpft sind.



Monitor



Fernsehgerät

Kanal 0 wird für die Auflistung von Eingaben und die Ausgabe von Fehlermeldungen benutzt, während Kanal 1 für die Programm- und Grafikausgabe und Kanal 2 für Programmauflistungen benutzt wird. Der Standardkanal kann mit einer wahlweisen Kanalangabe in dem entsprechenden Befehl geändert werden.

Hinweis

Der QL darf KEINESFALLS eingeschaltet werden, während eine Microdrive-Kassette eingelegt ist. Wenn ein Selbstladevorgang mit einer Microdrive-Kassette durchgeführt werden soll, so muß die Kassette zwischen dem Einschalten und der Betätigung von **F1** oder **F2** eingelegt werden.

STRING-TABELLEN STRING-VARIABLE

String-Tabellen und numerische Tabellen sind im Grunde genommen gleichartig, obwohl es bei der Verarbeitung durch SuperBASIC leichte Unterschiede gibt. Mit der letzten Dimension einer String-Tabelle wird die maximale Länge der Strings innerhalb der Tabelle definiert. String-Variablen können eine beliebige Länge von bis zu 32766 Zeichen aufweisen. Sowohl String-Tabellen als auch String-Variablen können *aufgeteilt* werden.

Die String-Längen auf der rechten und linken Seite einer String-Zuweisung brauchen nicht gleich zu sein. Sind die Größen nicht gleich, so wird der rechte String entsprechend abgeschnitten oder die Länge des linken String entsprechend verkürzt. Wird eine Zuweisung an einen aufgeteilten String vorgenommen, so wird gegebenenfalls das durch die Aufteilung entstandene "Loch" mit Leerzeichen aufgefüllt.

Die Dimension einer String-Tabelle braucht bei Zuweisungen nicht angegeben zu werden. Wird die Dimension nicht angegeben, so wird der ganze String ausgewählt, während durch Angabe eines einzelnen Elementes ein einzelnes Zeichen herausgenommen wird. Durch Angabe eines Teils wird ein Teilstring definiert.

Kommentar

Im Gegensatz zu vielen BASIC-Dialekten verarbeitet SuperBASIC String-Tabellen nicht als Strings mit fester Länge. Werden in einer String-Tabelle weniger Daten gespeichert als die Maximalgröße der String-Tabelle, so wird die Länge der Strings gekürzt.

Hinweis

Die Zuweisung von Daten an eine aufgeteilte String-Tabelle oder String-Variable kann unter Umständen nicht zu dem gewünschten Ergebnis führen. Die Länge des String wird durch derartige Zuweisungen nicht aktualisiert. Auf diese Weise ist es möglich, daß das System die Zuweisung nicht erkennt. Die Länge einer String-Tabelle oder einer String-Variablen wird nur aktualisiert, wenn dem ganzen String ein Wert zugewiesen wird.

Befehl	Funktion
FILL$	Erzeugt einen String
LEN$	Ermittelt die Länge eines String

STRING-VERGLEICH

Reihenfolge

. (Dezimalpunkt/Punkt)
Ziffern oder Zahlen in numerischer Reihenfolge.

AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz

Leerzeichen ! " # $ % & ' ( ) * + , – . / : ; < = > ? @ [\ ] ^ _ £ { | } ~ ©

Andere nicht ausdruckbare Zeichen.

Ein String kann in folgendem Verhältnis zu einem anderen String stehen:

Gleich: Sämtliche Zeichen oder Zahlen sind gleich oder gleichbedeutend.

Kleiner als: Der erste Teil des Strings, der sich von dem entsprechenden Zeichen in dem zweiten String unterscheidet, steht in der definierten Reihenfolge vor diesem.

Größer als: Der erste Teil des ersten Strings, der sich von dem entsprechenden Zeichen in dem zweiten String unterscheidet, steht in der definierten Reihenfolge hinter diesem.

Hier wird darauf hingewiesen, daß ein '.' bei einem Vergleich von Strings, die Zahlen enthalten (wie beispielsweise bei SuperBASIC-Vergleichen), als Dezimalpunkt behandelt werden kann. Außerdem wird darauf hingewiesen, daß der Vergleich von Strings mit nicht ausdruckbaren Zeichen zu unerwarteten Ergebnissen führen kann.

Vergleichsarten

Typ 0 – Zeichenweiser Vergleich, Groß- und Kleinschreibung werden berücksichtigt.
Typ 1 – Zeichenweiser Vergleich, Groß- und Kleinschreibung werden nicht berücksichtigt.
Typ 2 – Zahlen werden in numerischer Reihenfolge sortiert, Groß- und Kleinschreibung werden berücksichtigt.
Typ 3 – Zahlen werden in numerischer Reihenfolge sortiert, Groß- und Kleinschreibung werden nicht berücksichtigt.
Typ 0 – Wird normalerweise von dem SuperBASIC-System nicht benutzt.

Benutzung

Typ 1 – Datei- und Variablen-Vergleiche.
Typ 2 – SuperBASIC <, < =, =, > =, >, **INSTR** und < >
Typ 3 – SuperBASIC = = (Äquivalenz).

Hinweis

Weitere Hinweise zu den Vergleichsarten sind unter dem Begriff *Operatoren* zu finden

SYNTAX-DEFINITION

Die SuperBASIC-Syntax wird in einer Art "Meta-Sprache"beschrieben. Vier verschiedene Bezeichnungen werden benutzt:

| | | Eines der Elemente wird ausgewählt
[ ] Die enthaltenen Elemente werden nur bei Bedarf eingesetzt
* * Die enthaltenen Elemente können mehrfach eingesetzt werden.

.. Bereich
{ } Kommentar

Zum Beispiel:	| A | B |	A oder B
	[A]	A kann bei Bedarf eingesetzt werden
	* A *	A kann mehrfach eingesetzt werden
	A .. Z	A, B, C usw. bis Z

{Dies ist ein Kommentar}

Hier soll ein SuperBASIC-Name betrachtet werden:

Eine Folge von Zahlen, Ziffern und Unterstreichungszeichen, die mit einem Buchstaben beginnt und eventuell mit % oder $ endet.

Buchstabe:= |A .. Z
|a .. z

{Bei einem Buchstaben handelt es sich um:
ABCDEFGHIJKLMNOPQRSTUVWXYZ}
oder abcdefghijklmnopqrstuvwxyz

Ziffer:= |0|1|2|3|4|5|6|7|8|9|

{Bei einer Ziffer handelt es sich um 0 oder 1 oder 2 oder 3 oder 4 oder 5 oder 6 oder 7 oder 8 oder 9}

Unterstreichungszeichen:= _

{Ein Unterstreichungszeichen ist _}

Name = Buchstabe *[*Buchstabe* | *Ziffer* | *Unterstreichungszeichen*]* [| % | $ |]

Muß mit einem Buchstaben beginnen.

Eine Folge von Buchstaben, Ziffern und Unterstreichungszeichen, d.h. eine wahlweise Angabe kann wiederholt werden.

TABELLEN

Tabellen müssen stets mit **DIM** dimensioniert werden, bevor sie benutzt werden können. Ist eine Tabelle dimensioniert, so ist der Wert jedes der Tabellenelemente auf Null oder einen String mit einer Länge von Null gesetzt, sofern es sich um eine String-Tabelle handelt. Die Dimension einer Tabelle geht von Null bis zum angegebenen Wert. Die Anzahl definierter Dimensionen wird nur durch die Gesamtspeicherkapazität des Computers beschränkt. Eine Tabelle wird so gespeichert, daß der letzte definierte Index am schnellsten durchläuft.

Beispiel:

Die mit

```
DIMTabelle(2,4)
```

definierte Tabelle wird folgendermaßen gespeichert:

0,0	Anfangsadresse
0,1	
0,2	
0,3	
0,4	
1,0	
1,1	
1,2	
1,3	
1,4	
2,0	
2,1	
2,2	
2,3	
2,4	Endadresse

Das Element **Tabelle(a,b,c)** kann auch mit **Tabelle (a)(b)(c)** angesprochen werden.

Befehl	Funktion
DIM	Dimensioniert eine Tabelle
DIMN	Größe der einzelnen Dimensionen einer Tabelle

TON

Bei dem QL werden Töne durch den zweiten Prozessor des QL (einen 8049 Prozessor) erzeugt. Sie werden durch folgende Angaben gesteuert:

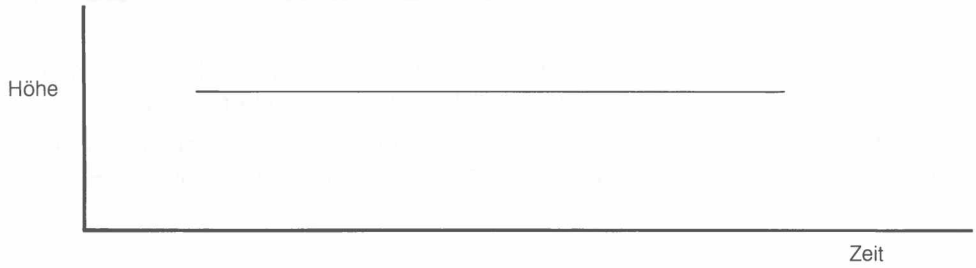
- bis zu zwei Tonhöhen
- Geschwindigkeit, mit der sich der Ton zwischen den Höhen bewegen soll, d. h. die Flanke;
- Verhalten des Tons nachdem eine der angegebenen Höhen erreicht ist, d. h. der Umlauf;
- Angabe, ob eine Zufälligkeit in den Ton eingebaut werden soll, d. h. Abweichungen von der Flanke;
- Angabe, ob Verfremdungen für den Ton aufgenommen werden sollen, Abweichungen bei jedem Tonzyklus.

Verfremdungen führen zu Summtönen, während die Zufälligkeiten, je nach anderen Parametern, zu "melodischen" Tönen oder Geräuschen führen.

Die Komplexität des Tons kann stufenweise aufgebaut werden, wobei nach und nach komplexere Töne entstehen. Durch diese Vorgehensweise werden Sie die Tonerzeugung mit dem QL am ehesten beherrschen.

STUFE 1

Hier wird nur eine Dauer und eine einzelne Tonhöhe angegeben. Die angegebene Höhe wird während der ganzen Zeit ausgegeben.

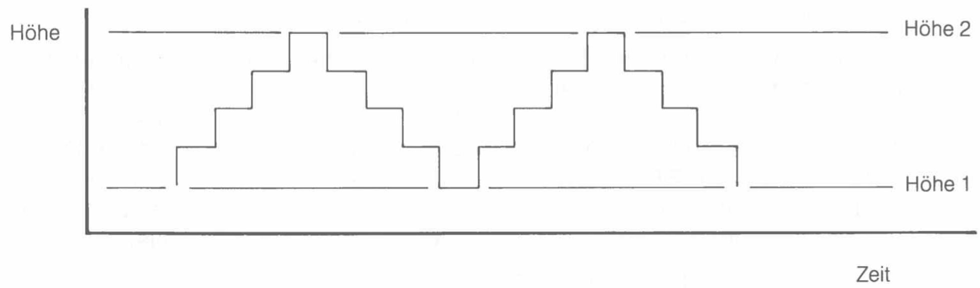


Dies ist der einfachste Ton-Befehl bei dem QL, mit Ausnahme des Befehls zur Beendigung des Tons.

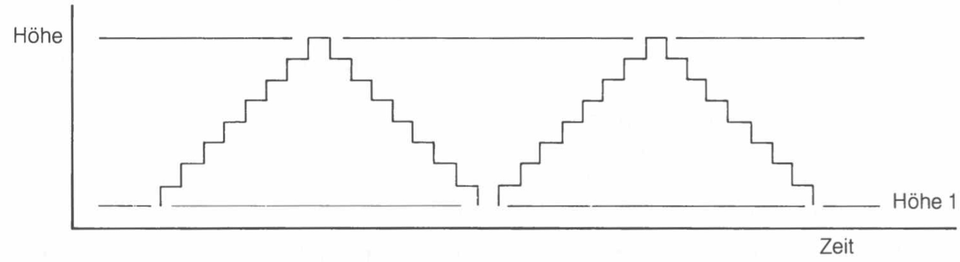
STUFE 2

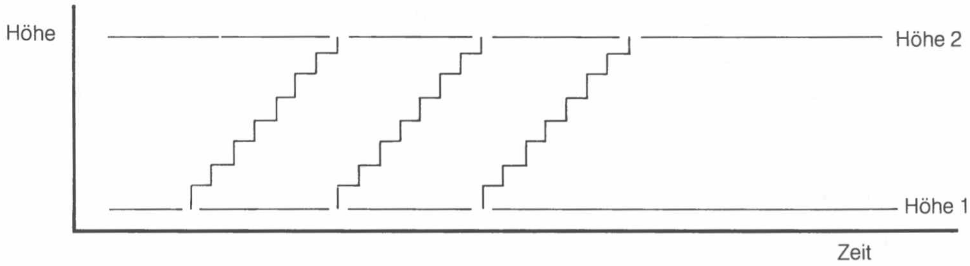
Zu dem Befehl kann eine zweite Höhe und der Gradient hinzugefügt werden. Danach schwingt der Ton zwischen den beiden Höhen mit einer von dem Gradienten angegebenen Geschwindigkeit hin und her.

Die auf dieser Stufe erzeugten Töne können zwischen melodischen Tönen, Brummtönen, Pfeiftönen und Jammergeräuschen variieren. Am besten wird mit den einzelnen Tönen experimentiert.



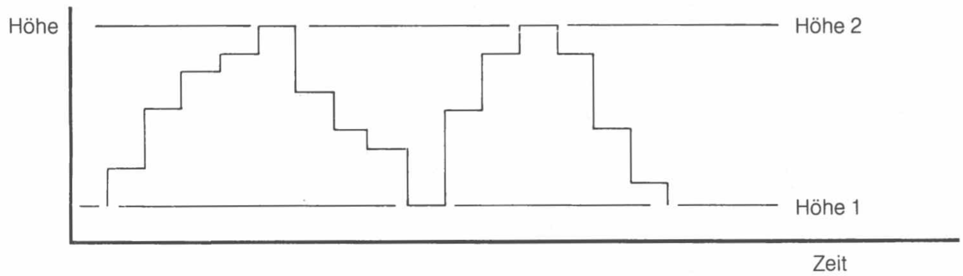
STUFE 3 Auf dieser Stufe kann ein Parameter hinzugefügt werden, der bestimmt, wie sich der Ton verhält, wenn er eine der angegebenen Höhen erreicht. Der Ton kann schwanken. Die Anzahl von Schwankungen kann angegeben werden, wobei er auch ständig hin- und herschwingen kann. Hier ist es noch wichtiger, daß experimentiert wird.





STUFE 4 Zu dem Ton kann eine Zufälligkeit hinzugefügt werden. Hier handelt es sich um eine Abweichung von dem angegebenen Schritt bzw. dem angegebenen Gradienten.

Je nach Zufälligkeit, die im Verhältnis zu der Tonhöhe und dem Gradienten hinzugefügt wird, wird ein sehr breites und unerwartetes Spektrum von Tönen erzeugt.



STUFE 5 Nun kann noch eine weitere Variation hinzugefügt werden, indem eine "Verfremdung" angegeben wird. Mit ihr wird der Höhe kontinuierlich ein Zufallsfaktor hinzugefügt. Durch die Verfremdung wird der Ton häufig zu einem Summton.

Durch Kombination aller obigen Effekte kann ein sehr breites Tonspektrum erzeugt werden, wobei es zu vielen unerwarteten Tönen kommt. Mit den QL-Tönen wird am besten experimentiert, bevor sie endgültig festgelegt werden. Wird eine Zeitdauer von Null angegeben, so kann der Ton ständig wiederholt werden. Auf diese Weise kann mit einer Folge von **BEEP**-Befehlen experimentiert werden bis der gewünschte Ton erzeugt wird. Hier jedoch eine Warnung: schon kleine Änderungen beim Wert eines einzigen Parameters können sich verhängnisvoll auf den Ton auswirken.

Beispiel: `BEEP 0,1,1,4500,0,5,0,9`

TURTLE-GRAFIK

SuperBASIC verfügt über eine Reihe von Befehlen für Turtle-Grafiken:

Befehl	Funktion
PENUP	Stoppt das Zeichnen
PENDOWN	Startet das Zeichnen
MOVE	Verschiebt die Turtle
TURN	Dreht die Turtle
TURNTO	Dreht in eine bestimmte Richtung

Diese Befehlsgruppe stellt das Minimum dar. Sie wird normalerweise zusammen mit anderen Prozedur benutzt, um die Befehle zu erweitern. Zum Beispiel:

```
100 DEFine PROCedure vor(strecke)
110   MOVE strecke
120 END DEFine
130 DEFine PROCedure umkehr(strecke)
140   MOVE -strecke
150 END DEFine
160 DEFine PROCedure links(winkel)
170   TURN winkel
180 END DEFine
190 DEFine PROCedure rechts(winkel)
200   TURN -winkel
210 END DEFine
```

Damit werden einige der bekannteren Befehle für Turtle-Grafiken definiert.

Standardmäßig ist der Turtle-Stift abgehoben und die Turtle zeigt in Richtung 0°, d. h. zur rechten Seite des Fensters.

Der **FILL**-Befehl kann auch mit Abbildungen benutzt werden, die mit Turtle-Grafiken gezeichnet werden. Außerdem können normale Grafiken und Turtle-Grafiken gemischt werden, auch wenn die Richtung der Turtle durch die normalen Grafik-Befehle nicht geändert wird.

UHR

Der QL enthält eine Echtzeituhr, die läuft, sobald der Computer eingeschalter wird.

Für Datum und Uhrzeit wird das ISO-Standardformat benutzt:

```
1983 JAN 01 12:09:10
```

Aktuelle Angaben für Jahr, Monat, Tag und Uhrzeit können durch **DATE$** abgerufen werden.

Befehl	Funktion
SDATE	Stellt die Uhr ein
ADATE	Ändert die Uhr
DATE	Gibt das Datum in Form einer Zahl zurück
DATE$	Gibt das Datum und die Zeit in Form eines Strings zurück
DAY$	Gibt den Wochentag zurück.

UNTER-BRECHUNG

Reagiert der Computer zu irgendeinem Zeitpunkt nicht richtig oder möchten Sie ein SuperBASIC-Programm bzw. einen SuperBASIC-Befehl stoppen, so gehen Sie folgendermaßen vor:

Halten Sie

CTRL

gedrückt und betätigen Sie die

LEERTASTE

Ein auf diese Weise unterbrochenes Programm kann mit dem **CONTINUE**-Befehl fortgesetzt werden.

Sie können die Bildschirmausgabe eines Programmes anhalten, wie z. B. bei Print, List und Grafikprozeduren.

Halten Sie dazu

CTRL

gedrückt und betätigen Sie die Taste F5.

Nach Druck der Leertaste wird das Programm fortgesetzt.

ZEICHEN-VORRAT UND TASTEN

Dezimal	Hex	Tasten	Anzeige/Funktion
0	00	CTRL ESC	NULL
1	01	CTRL A	
2	02	CTRL B	
3	03	CTRL C	Eingabekanal umschalten
4	04	CTRL D	
5	05	CTRL E	
6	06	CTRL F	
7	07	CTRL G	
8	08	CTRL H	
9	09	TAB (CTRL I)	Tabulator
10	0A	ENTER (CTRL J)	Neue Zeile/Befehlseingabe
11	0B	CTRL K	
12	0C	CTRL L	
13	0D	CTRL M	Enter
14	0E	CTRL N	
15	0F	CTRL O	
16	10	CTRL P	
17	11	CTRL Q	
18	12	CTRL R	
19	13	CTRL S	
20	14	CTRL T	
21	15	CTRL U	
22	16	CTRL V	
23	17	CTRL W	
24	18	CTRL X	
25	19	CTRL Z	
26	1A	CTRL Y	
27	1B	ESC (CTRL SHIFT Ü)	Befehlsabbruch
28	1C	CTRL SHIFT <	
29	1D	CTRL SHIFT +	
30	1E	CTRL SHIFT	
31	1F	CTRL SHIFT ESC	
32	20	Leerzeichen	Leerzeichen
33	21	SHIFT 1	!
34	22	SHIFT 2	"
35	23	#	#
36	24	SHIFT 4	$
37	25	SHIFT 5	%
38	26	SHIFT 6	&
39	27	SHIFT #	'
40	28	SHIFT 8	(
41	29	SHIFT 9	)
42	2A	SHIFT +	*
43	2B	+	+
44	2C	,	,
45	2D	–	–
46	2E	.	.
47	2F	SHIFT 7	/

Dezimal	Hex	Tasten	Anzeige/Funktion
48	30	0	0
49	31	1	1
50	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	SHIFT .	:
59	3B	SHIFT ,	;
60	3C	<	<
61	3D	SHIFT 0	=
62	3E	SHIFT <	>
63	3F	SHIFT ß	?
64	40	CTRL	@
65	41	SHIFT A	A
66	42	SHIFT B	B
67	43	SHIFT C	C
68	44	SHIFT D	D
69	45	SHIFT E	E
70	46	SHIFT F	F
71	47	SHIFT G	G
72	48	SHIFT H	H
73	49	SHIFT I	I
74	4A	SHIFT J	J
75	4B	SHIFT K	K
76	4C	SHIFT L	L
77	4D	SHIFT M	M
78	4E	SHIFT N	N
79	4F	SHIFT O	O
80	50	SHIFT P	P
81	51	SHIFT Q	Q
82	52	SHIFT R	R
83	53	SHIFT S	S
84	54	SHIFT T	T
85	55	SHIFT U	U
86	56	SHIFT V	V
87	57	SHIFT W	W
88	58	SHIFT X	X
89	59	SHIFT Y	Y
90	5A	SHIFT Z	Z
91	5B	CTRL 9	[
92	5C		
93	5D	CTRL 0	]
94	5E	SHIFT \	^
95	5F	SHIFT –	_
96	60	CTRL 7	£
97	61	A	a
98	62	B	b
99	63	C	c
100	64	D	d
101	65	E	e
102	66	F	f
103	67	G	g
104	68	H	h
105	69	I	i
106	6A	J	j
107	6B	K	k
108	6C	L	l
109	6D	M	m
110	6E	N	n
111	6F	O	o

Dezimal	Hex	Tasten	Anzeige/Funktion
112	70	P	p
113	71	Q	q
114	72	R	r
115	73	S	s
116	74	T	t
117	75	U	u
118	76	V	v
119	77	W	w
120	78	X	x
121	79	Y	y
122	7A	Z	z
123	7B	CTRL ß	{
124	7C	CTRL 8	|
125	7D	CTRL #	}
126	7E	CTRL <	~
127	7F	SHIFT ESC	©
128	80	Ä	ä
129	81	CTRL SHIFT 1	a tilde
130	82	CTRL SHIFT Ä	a circle
131	83	CTRL SHIFT 3	e acute
132	84	Ö	ö
133	85	CTRL SHIFT 5	o tilde
134	86	CTRL SHIFT 7	o bar
135	87	Ü	ü
136	88	CTRL SHIFT 9	c cedilla
137	89	CTRL SHIFT 0	n tilde
138	8A	CTRL SHIFT 8	ae diphthong
139	8B	CTRL SHIFT #	oe diphthong
140	8C	CTRL ,	a acute
141	8D	CTRL SHIFT 4	a grave
142	8E	CTRL .	a circumflex
143	8F	CTRL –	e umlaut
144	90	CTRL SHIFT V	e grave
145	91	CTRL 1	e circumflex
146	92	CTRL 2	i umlaut
147	93	CTRL 3	i acute
148	94	CTRL 4	i grave
149	95	CTRL 5	i circumflex
150	96	CTRL 6	o acute
151	97	CTRL SHIFT ,	o grave
152	98	CTRL SHIFT D	o circumflex
153	99	CTRL Ä	u acute
154	9A	CTRL SHIFT Ö	u grave
155	9B	CTRL Ö	u circumflex
156	9C	ß	ß
157	9D	CTRL SHIFT G	cent symbol
158	9E	CTRL SHIFT .	Yen symbol
159	9F	CTRL SHIFT –	backquote
160	A0	SHIFT Ä	Ä
161	A1	CTRL SHIFT A	A tilde
162	A2	CTRL SHIFT B	A circle
163	A3	CTRL SHIFT C	E acute
164	A4	SHIFT Ö	Ö
165	A5	CTRL SHIFT E	O tilde
166	A6	CTRL SHIFT F	O bar
167	A7	SHIFT Ü	Ü
168	A8	CTRL SHIFT H	C cedilla
169	A9	CTRL SHIFT I	N tilde
170	AA	CTRL SHIFT J	AE diphthong
171	AB	CTRL SHIFT K	OE diphthong
172	AC	CTRL SHIFT L	alpha
173	AD	CTRL SHIFT M	delta
174	AE	CTRL SHIFT N	theta
175	AF	CTRL SHIFT O	lambda

Dezimal	Hex	Tasten	Anzeige/Funktion
176	B0	CTRL SHIFT P	mu
177	B1	CTRL SHIFT Q	pi
178	B2	CTRL SHIFT R	phi
179	B3	CTRL SHIFT S	Spanish inverse ! mark
180	B4	CTRL SHIFT T	Spanish inverse ? mark
181	B5	CTRL SHIFT U	Swedish script mark
182	B6	SHIFT 3	Section symbol
183	B7	CTRL SHIFT W	Continental cross/circle
184	B8	CTRL SHIFT X	≪
185	B9	CTRL SHIFT z	≫
186	BA	CTRL SHIFT Y	degree
187	BB	CTRL Ü	division symbol
188	BC	CTRL SHIFT 2	←
189	BD	CTRL +	→
190	BE	CTRL SHIFT 6	↑
191	BF	CTRL SHIFT ß	↓
192	C0	←	
193	C1	ALT ←	
194	C2	CTRL ←	
195	C3	CTRL ALT ←	
196	C4	SHIFT ←	
197	C5	SHIFT ALT ←	
198	C6	SHIFT CTRL ←	
199	C7	SHIFT CTRL ALT ←	
200	C8	→	
201	C9	ALT →	
202	CA	CTRL →	
203	CB	CTRL ALT →	
204	CC	SHIFT →	
205	CD	SHIFT ALT →	
206	CE	SHIFT CTRL →	
207	CF	SHIFT CTRL ALT →	
208	D0	↑	
209	D1	ALT ↑	
210	D2	CTRL ↑	
211	D3	CTRL ALT ↑	
212	D4	SHIFT ↑	
213	D5	SHIFT ALT ↑	
214	D6	SHIFT CTRL ↑	
215	D7	SHIFT CTRL ALT ↑	
216	D8	↓	
217	D9	ALT ↓	
218	DA	CTRL ↓	
219	DB	CTRL ALT ↓	
220	DC	SHIFT ↓	
221	DD	SHIFT ALT ↓	
222	DE	SHIFT CTRL ↓	
223	DF	SHIFT CTRL ALT ↓	
224	E0	↕	
225	E1	ALT ↕	
226	E2	CTRL ↕	
227	E3	ALT CTRL ↕	
228	E4	SHIFT ↕	
229	E5	SHIFT ALT ↕	
230	E6	SHIFT CTRL ↕	
231	E7	SHIFT CTRL ALT ↕	
232	E8	F1	
233	E9	CTRL F1	
234	EA	SHIFT F1	
235	EB	CTRL SHIFT F1	
236	EC	F2	
237	ED	CTRL F2	
238	EE	SHIFT F2	
239	EF	CTRL SHIFT F2	

Dezimal	Hex	Tasten	Anzeige/Funktion
240	F0	F3	
241	F1	CTRL F3	
242	F2	SHIFT F3	
243	F3	CTRL SHIFT F3	
244	F4	F4	
245	F5	CTRL F4	
246	F6	SHIFT F4	
247	F7	CTRL SHIFT F4	
248	F8	F5	
249	F9	CTRL F5	
250	FA	SHIFT F5	
251	FB	CTRL SHIFT F5	
252	FC	SHIFT Leerzeichen	
253	FD	SHIFT TAB	
254	FE	SHIFT ENTER	
255	FF	Siehe unten	

Wenn die ALT-Taste mit einer beliebigen anderen Taste, außer den Cursor-Steuertasten oder der UMSCHALT-Taste betätigt wird, wird der Code FF erzeugt, gefolgt von einem Byte, das anzeigt, was der Tastencode gewesen wäre, wenn die ALT-Taste nicht gedrückt wäre.

Zeichen bis 20 Hex sind Kontrollzeichen oder nicht druckbare Zeichen. Alternative Zeichen sind in Klammern gesetzt.

CTRL-C wird von Qdos abgefangen und kann nicht erkannt werden ohne Änderung der System Variablen.

Die Zeichen zwischen C0 und DF sind Curser-Kontroll Kommandos.

Die UMSCHALT-Taste (Caps Lock) und CTRL-F5 werden von Qdos abgefangen und können nur mit spezieller Software erkannt werden.

A
ANWEISUNG 1
direkter Befehl 1,13
AUFTEILUNG
Tabelle 2
Teiltabellen 2
AUSDRÜCKE
bedingte 3
numerische 3
String 3

B
BASIC 4
Baudrate 10
Bedingte Ausdrücke 3
BEEP 51
BEFEHLE 5
Prozeduren 5
BILDSCHIRM
hochauflösend 6
niedrigauflösend 6

C
Cartridges
ROM 41
Microodrive 29
Circle 22
Code 7
Console 14
CTS 10

D
DATEITYPEN, DATEIEN
Daten 7
Exec 7
Code 7
DATENTYPENUMWANDLUNG 8
Operatoren 8
DATENTYPEN, VARIABLE
ganze Zahlen 9
Gleitkommazahlen 9
String 9
Name 9
DATENÜBERTRAGUNG, RS-232-C 10
Baudrate 10
DIREKTER BEFEHL 1,13

E
Ein-/Ausgabe 39
EINHEITEN 14
Konsoleinheit 14
Bildschrimausgabe 15
Serieller Anschluß 15
Netzwerk 15
Microdrive 15
Exec 7

F
FARBE 16
Vollfarben 16
Punktmuster 16
FEHLERBEHANDLUNG 18
Fehlerbehebung 19
FENSTER 20
Formatieren 29
FUNKTIONEN UND PROZEDUREN 21

G
Ganze Zahlen 9
Gleitkommazahlen 9
GRAFIK 22
grafisches Koordinatensystem 22
Koordinatenursprung 22
Grafik-Cursor 23

H
Handshaking 10
hochauflösender Bildschirm 6

I
Initialisierung 1

J
JOYSTICK 24

K
KANÄLE 25
Konsole 14
KOORDINATEN,
GRAFIK-KOORDINATENSYSTEM 26
Pixel 26
Koordinatenursprung 26
Pixel-Koordinatensystem 27
Ausgabe am Bildschirmfenster 27

L
Local variable 21

M
MATHEMATISCHE FUNKTIONEN 28
MICRODRIVES 29
Formatieren 29
MONITOR 31
RGB-Anschluß 31
Multitasking 40

N
NAME 32
NETZWERK 33
Stationsnummer 33

niedrigauflösender Bildschirm ... 6
numerische Ausdrücke ... 3

O

Operating System ... 38
OPERATOREN ... 34
  Priorität ... 34

P

Parameter ... 21
PERIPHERIE-ERWEITERUNG ... 35
Pixel ... 26
Pixel-Koordinatensysteme ... 27
Priorität ... 34
PROGRAMM ... 37
  Anweisungen ... 37

Q

QDOS ... 38
  Speichereinteilung ... 38
  Systemaufrufe ... 39
  Ein-/Ausgabe ... 39
  Multitasking ... 40
  Einheiten ... 40

R

RGB-Anschluß ... 31
ROM-KASSETTENANSCHLUSS ... 41
RS-232-C ... 10

S

Serieller Anschluß ... 15
SCHLEIFEN ... 42
SPEICHEREINTEILUNG ... 43
Standartbildschirm ... 46
STANDARTWERTE ... 45
STARTEN ... 46
  Standartbildschirm ... 46
String ... 2,13
STRING-TABELLEN, STRING-VARIABLE ... 47
STRING-VERGLEICH ... 48
  Vergleichsart ... 48
SYNTAX-DEFINITION ... 49
Systemaufruf ... 39

T

TABELLEN ... 50
Tasten ... 56
TON ... 51
TURTLE-GRAFIK ... 52

U

UHR ... 54
UNTERBRECHUNG ... 55

V

Variable ... 9
  Locale ... 21

W

Window ... 52

Z

ZEICHENVORRAT UND TASTEN ... 56

sinclair

QL

QL Quill

WICHTIG

BEVOR SIE DIE APPLIKATIONEN BENUTZEN, LESEN SIE BITTE DIE NACHSTEHENDE BESCHREIBUNG, UM ARBEITSKOPIEN ZU ERSTELLEN.

ARBEITSKOPIEN

Fertigen Sie bitte immer mindestens eine Arbeitskopie an, bevor Sie eines der vier QL Programme benutzen. Arbeiten Sie dann immer nur mit der Arbeitskopie. Bewahren Sie die Originalkassetten mit den Programmen sicher auf. Benutzen Sie die Originalkassetten nur, um bei Bedarf Kopien anzufertigen. So schützen Sie sich vor dem Verlust Ihrer Programme.

Bei allen magnetischen Speichermedien, also auch bei Microdrive-Kassetten, kann bei Gebrauch Datenverlust entstehen. Daher sollten Sie alle Programme und Daten, die Sie noch benötigen, mindestens zweimal speichern. Fertigen Sie sich deshalb immer Sicherungskopien an. So schützen Sie sich vor Datenverlust. Immer, wenn Sie einer Kassette Daten oder Programme hinzufügen, sollten Sie das auch bei der Sicherungskopie tun. Wenn Sie das versäumen, laufen Sie Gefahr, alles, was Sie seit der letzten Datensicherung hinzugefügt haben, zu verlieren. Hinweise zur Anfertigung von Sicherungskopien finden Sie im Abschnitt *Information*.

by Dick de Grandis-Harrison *(Psion Limited)*

KAPITEL 1
WAS IST QL QUILL?

QL Quill ist ein äußerst komfortables Textverarbeitungsprogramm. Konzipiert für optimale Leistungsfähigkeit und Flexibilität, ist es dennoch einfach im Gebrauch. Wie Sie bald sehen werden, hält Sie das System ständig auf dem laufenden.

Alles, was Sie bisher mit der Schreibmaschine erledigt haben, können Sie von nun an Ihrem Textverarbeitungssystem überlassen. Von der Funktion her sind sich die beiden Geräte ganz ähnlich, doch sind Textverarbeitungssysteme den herkömmlichen Schreibmaschinen in vielen Dingen überlegen. Der vielleicht überzeugendste Vorteil ist die schnelle und unkomplizierte Fehlerkorrektur. Da der eingetippte Text nicht sofort auf Papier ausgedruckt wird, können Sie am Bildschirm beliebig viele Änderungen und Korrekturen vornehmen und mit dem Drucken so lange warten, bis Sie mit dem Ergebnis vollauf zufrieden sind.

Im Verlauf dieses Einführungskurses werden Sie noch eine Menge weiterer Vorteile kennenlernen. So brauchen Sie beispielsweise nicht nach jeder Zeile den Wagenrücklauf zu betätigen. Quill erkennt das Zeilenende von selbst und fügt automatisch eine Zeilenschaltung ein, sobald der rechte Textrand erreicht ist. **ENTER** brauchen Sie immer nur dann zu drücken, wenn Sie einen neuen Absatz beginnen wollen. Bei jedem Zeilenumbruch erfolgt ein automatischer Randausgleich, d.h. der Text wird links- und rechtsbündig justiert. Diese Art der Darstellung, die man *Blocksatz* nennt, verleiht dem Text ein professionelles Aussehen – ganz ohne Ihr Zutun. Wie die meisten Quill-Funktionen kann auch die Art der Textjustierung individuellen Ansprüchen angepaßt werden.

In Situationen, wo Sie nicht ganz sicher sind, wie es weitergeht, holen Sie sich mit der Funktionstaste **F1** schnell Hilfe. Irrtümlich angeforderte, noch nicht ausgeführte Operationen, etwa einen Befehl, können Sie jederzeit mit **ESC** rückgängig machen.

KAPITEL 2
FANGEN SIE AN

QL QUILL LADEN

Laden Sie QL Quill gemäß der Anleitung in der Einführung zu den QL-Programmen. Nach abgeschlossener Ladeprozedur meldet sich das System mit dieser Anzeige:

```
LADEN DER QL QUILL
Textverarbeitung
Version x.y
Copyright ©1984 PSION Ltd.
Alle Rechte vorbehalten
```

wobei x.y die Versions-Nummer bedeutet, z.B. 2.23.

Auf die Kassette im Microdrive 1 wird beim Ausdrucken eines Dokuments und beim Aufrufen der Hilfe-Datei zugegriffen.

Solange der eingegebene Text nicht länger als etwa 3 Seiten ist, kommt Quill ohne eine Kassette im Microdrive 2 aus. Bei Bedarf fordert es eine zweite Kassette an. Schieben Sie diese sorgfältig ein und entfernen Sie sie erst, wenn das Dokument entweder gespeichert oder als wertlos aufgegeben wurde.

HILFE F1 DIALOG F2	CURSOR ↑ ← → ↓	TEXT Neuer Absatz: ↵ Löschen: CTRL & ←↑↓→ Modus ändern: SHIFT + F4	TEXTSTIL F4	BEFEHLE F3 ABBRUCH ESC

```
....:....1....:....2....:....3....:....4....:....5....:....6....:....7....:....8

MODUS: EINFÜGEN          WORTE: 0          ZEILE: 1            SEITE: 1
TEXTSTIL:Normal                            DOKUMENT: kein Name
```

Abbildung 2.1 Die Hauptanzeige auf einem Monitor (80 Zeichen pro Zeile)

ANZEIGEFORMAT

Zu Beginn sollte auf Ihrem Bildschirm eine Anzeige gemäß Abb. 2.1 bzw. Abb. 2.2 erscheinen. Man nennt dies die *Hauptanzeige*.

Quill verfügt über drei verschiedene Formate: 80, 64 oder 40 Zeichen pro Zeile. Das Bild auf handelsüblichen Fernsehgeräten ist oft nicht scharf genug für das 80er Format, und es empfiehlt sich, die Anzeige mit 64 oder 40 Zeichen zu wählen. Der Bildschirmaufbau beim 80er und 64er Format ist sehr ähnlich; etwas anders ist er nur bei der 40-spaltigen Anzeige. Dort präsentiert sich die Hauptanzeige gemäß Abb. 2.2.

Je nachdem, ob Sie zu Anfang **F1** oder **F2** drücken, wählt Quill das Bildschirmformat 80 bzw. 64, doch können Sie diese Einstellung jederzeit mit Hilfe des FORMAT-Befehls ändern, den wir noch ausführlicher besprechen werden.

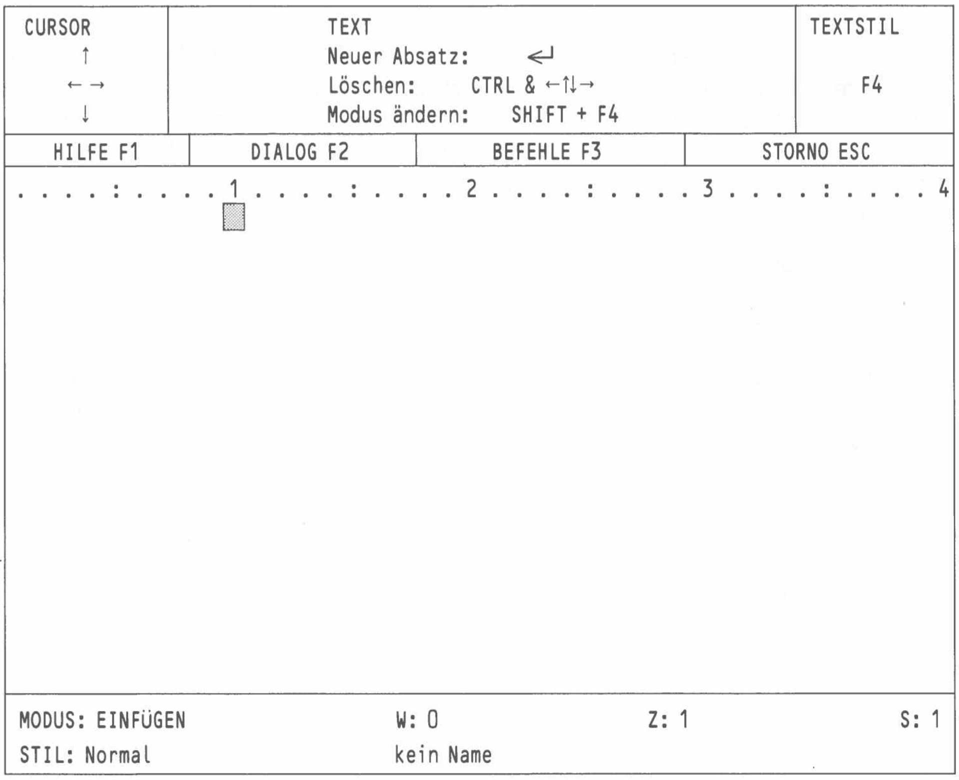


Abbildung 2.2 Die Hauptanzeige bei 40 Zeichen pro Zeile

Abgesehen vom optischen Unterschied in der Bildschirmanzeige funktioniert Quill mit allen drei Formaten genau gleich. Die Mehrzahl der Abbildungen zeigt das Format mit 80 Spalten.

Der Bildschirm teilt sich in drei Hauptzonen auf: den *Anzeigebereich* in der Mitte, den *Statusbereich* unten und den *Steuerbereich* oben.

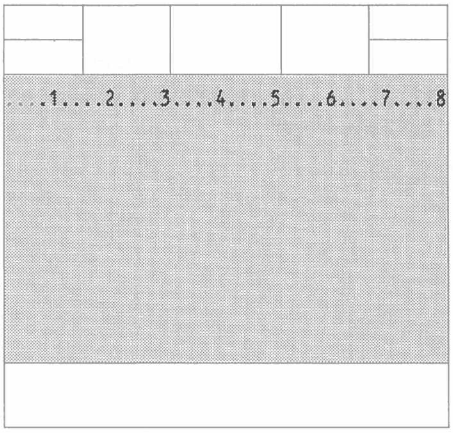


Abbildung 2.3 Der Anzeigebereich

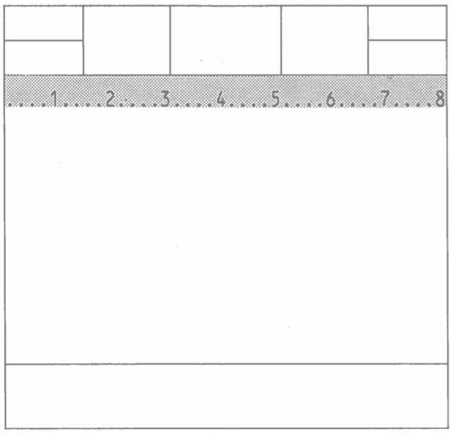


Abbildung 2.4 Die Spaltenskala

Der Anzeigebereich

Der größte Bereich im Zentrum des Bildschirms ist für den Text Ihres Dokuments reserviert. Fast alles, was Sie über die Tastatur eingeben, erscheint hier.

Die oberste Zeile im Anzeigebereich ist die *Spaltenskala*, auch *Lineal* genannt. Jeder Punkt markiert eine Zeichenposition, wobei jeweils jede fünfte Stelle mit einem Doppelpunkt gekennzeichnet ist und jede zehnte mit einer Ziffer.

Der Statusbereich

Der *Statusbereich*, der die untersten drei Zeilen des Bildschirms umfaßt, enthält Informationen zum aktuellen Dokument, wie z.B. seinen Namen. Solange Sie ein Dokument nicht benennen, steht in dieser Rubrik der Vermerk "kein Name".

Ferner entnehmen Sie der Statuszeile, daß sich Quill zur Zeit im Modus *Einfügen* befindet. Mit anderen Worten: Ihre Texteingabe wird in das Dokument eingefügt und überschreibt nicht etwa bereits bestehenden Text. Der Vermerk "Normal" besagt, daß keine besondere Schriftart gewählt wurde, sondern im normalen *Stil* geschrieben wird. Auf Wunsch können Sie aber auf Fettdruck, Unterstreichung, Hoch- und Tiefstellung umschalten. Wie dabei vorzugehen ist, erklären wir etwas später.

Weitere Informationen im Statusbereich sind die Anzahl der Wörter im aktuellen Dokument und die momentane Cursorposition (Zeile, Seite). Bei Beginn der Textverarbeitung befindet sich der Cursor auf Zeile 1, Seite 1 eines Dokuments, das 0 Worte enthält.

Ebenfalls im Statusbereich erscheinen im Zusammenhang mit *Befehlen* eingegebene Texte und Meldungen. (Aufgerufen werden Befehle mit Hilfe der Funktionstaste **F3**). Beim Befehl **Suchen** (vgl. Kapitel 5) z.B. werden der gewünschte Suchbegriff und die Nachfragen im Statusbereich angezeigt.

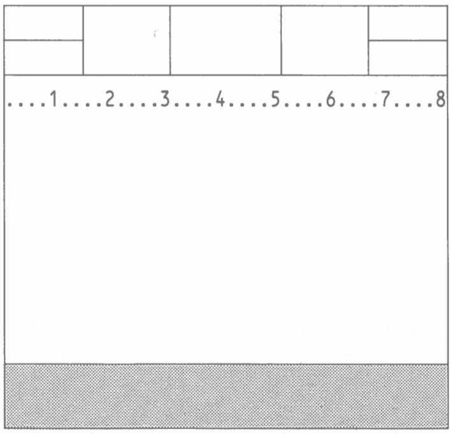


Abbildung 2.5 Der Statusbereich

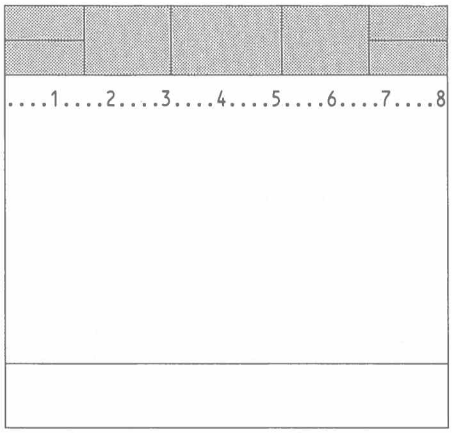


Abbildung 2.6 Der Steuerbereich

Der Steuerbereich

Der Steuerbereich ist der Streifen am oberen Rand des Bildschirms. Hier werden die allgemeingültigen Optionen für Hilfe (**F1**), zum Ein- und Ausblenden der Dialogmeldungen (**F2**), zum Aufrufen der Befehle (**F3**) und zum Abbrechen von Operationen (**ESC**) angezeigt. Außerdem erkennen Sie drei Quill-spezifische Funktionen, welche die drei mittleren Felder des Streifens belegen, nämlich:

CURSOR	Cursorsteuerung mit den Pfeiltasten
TEXT	Text anhängen oder entfernen
TEXTSTIL	Schriftart ändern.

DER CURSOR

Auf der ersten Textzeile unter der Spaltenskala sitzt ein kleiner roter Block. Das ist der *Cursor*, der die Position markiert, an der das nächste eingegebene Zeichen erscheint.

Die Steuerung des Cursors innerhalb des Arbeitsbereichs erfolgt mit den vier Pfeiltasten, wie im CURSOR-Feld angezeigt. Allerdings kann sich der Cursor nur im bereits gefüllten Arbeitsbereich bewegen, wobei er bei Fingerdruck auf eine Pfeiltaste um einen Schritt in der betreffenden Richtung springt. Er kann nicht über das Textende hinausspringen, und solange Ihr Dokument leer ist, sitzt er in seiner ursprünglichen Position fest.

Innerhalb des Textes kann der Cursor auch größere Sprünge machen. Durch gleichzeitiges Festhalten von **SHIFT** und Drücken des Rechts- oder Linkspfeils vergrößert sich die Sprungweite auf jeweils ein Wort nach rechts oder nach links. Gleichzeitiges Drücken von **SHIFT** und dem Auf- bzw. dem Abwärtspfeil versetzt den Cursor auf den vorherigen bzw. den nächsten Absatzanfang.

TEXT

Die Option TEXT im Mittelfeld des Steuerbereichs zeigt die verschiedenen Möglichkeiten zur Textbearbeitung. Bei der normalen Texteingabe über die Tastatur erscheint jedes Zeichen an der Cursorposition.

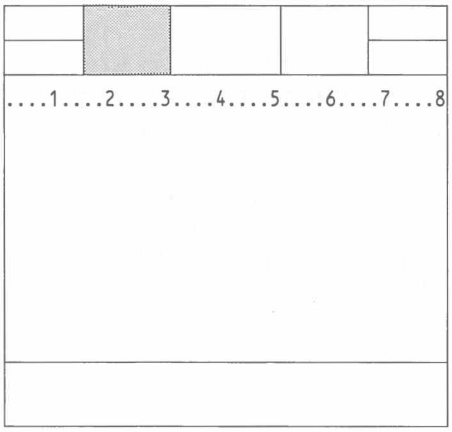


Abbildung 2.7 Die Cursorsteuerung

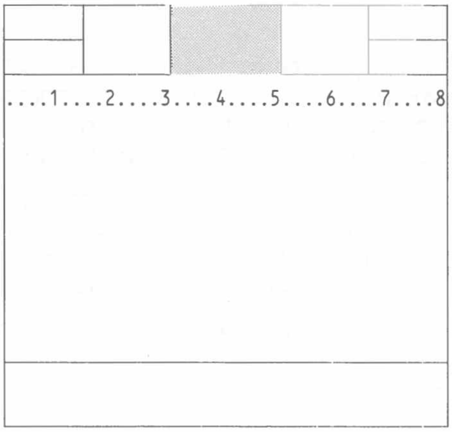


Abbildung 2.8 Die Text-Optionen

Die zweite Zeile im Feld TEXT zeigt, daß nach Drücken von **ENTER** ein neuer Absatz beginnt. Bei fortlaufendem Text darf also die Entertaste nicht betätigt werden. Sie brauchen auch gar nicht auf das Zeilenende zu achten, weil Quill die Zeilenschaltung automatisch vornimmt. Sie können am Bildschirm mitverfolgen, wie das Wort, welches nicht mehr auf die laufende Zeile paßt, auf die nächste übernommen wird, während die Wortzwischenräume vergrößert werden, um einen perfekten Randausgleich (*Blocksatz*) zu erzielen.

Probieren Sie diese Textoption gleich einmal aus, indem Sie **ENTER** und danach einige Buchstabentasten drücken. Machen Sie sich keine Sorgen, wenn die Einrückung zur Markierung eines neuen Textabsatzes nicht so ist, wie Sie es gern haben wollen. Wie sie zu ändern ist, erfahren Sie in Kapitel 4.

Es ist auch möglich, Zeichen in ein Dokument einzufügen, die auf der Tastatur nicht angegeben sind. Erzeugt werden sie unter Zuhilfenahme der Tasten **CTRL** bzw. **CTRL** und **SHIFT** in Kombination mit einer anderen Taste.

Der Abschnitt *Begriffe* in diesem Handbuch enthält eine vollständige Liste aller verfügbaren Zeichen, zusammen mit den Eingabevorschriften.

Um ein Zeichen links oder rechts von der Cursorposition zu löschen, halten Sie **CTRL** fest, während Sie kurz den Links– oder Rechtspfeil anschlagen.

Vielleicht ist Ihnen bei der Texteingabe aufgefallen, daß sich im Statusbereich einiges tut: Die Wort– und Zeilenzählung wird laufend auf den neuesten Stand gebracht. Die übrigen Angaben bleiben unverändert. Das Dokument ist nach wie vor namenlos, da Sie ihm erst dann einen Namen zuweisen, wenn Sie es auf einer Microdrive-Kassette speichern (Beschreibung in Kapitel 7).

Jetzt, da Ihr Dokument bereits etwas Text enthält, können Sie mit den Cursorsteuertasten experimentieren. Plazieren Sie den Cursor anschließend auf das Textende.

TEXTSTIL

Eine weitere Option im Steuerbereich wird mit TEXTSTIL bezeichnet. Sie bestimmt das Aussehen des Textes.

Mit **F4** holen Sie sich fünf Möglichkeiten zur Auswahl auf den Bildschirm:

Umstellen auf **F**ettschrift
Hochstellen von Zeichen
Tiefstellen von Zeichen
Unterstreichungsfunktion einschalten
Übermalen von bestehendem Text.

Alle diese Funktionen werden durch einmaliges Drücken von **F4**, gefolgt vom betreffenden Anfangsbuchstaben im TEXTSTIL-Feld, gestartet. Probieren wir es doch gleich einmal mit der Unterstreichungsfunktion aus. Erst **F4** drücken, dann die **U**-Taste.

Sie haben wiederum die übliche Bildschirmmaske vor sich, doch wenn Sie genau hinschauen, bemerken Sie in der Rubrik STIL jetzt die Angabe "Unterstr.". In diesem Zustand eingegebener Text wird unterstrichen.

Quill zeigt Ihnen am Bildschirm, was später aufs Papier kommt. Die einzigen Ausnahmen sind der obere und der untere Seitenrand und der Zeilenabstand, wenn dieser doppelt oder dreifach gewählt wird. Auf die Darstellung wird in diesen Fällen verzichtet, weil sie den Bildschirminhalt erheblich reduzieren würde.

Zum Ausschalten der Unterstreichung wird ein zweites Mal **F4** und **U** gedrückt. Wie Sie sehen, wird diese Funktion wie ein Ein/Aus–Kippschalter betätigt. Ob es auch wirklich geklappt hat, können Sie leicht nachprüfen, indem Sie ein paar Zeichen eintippen. Außerdem erscheint im Statusbereich wiederum der Vermerk "Normal".

Für eine ausführliche Beschreibung der Textstil-Optionen verweisen wir Sie auf Kapitel 4.

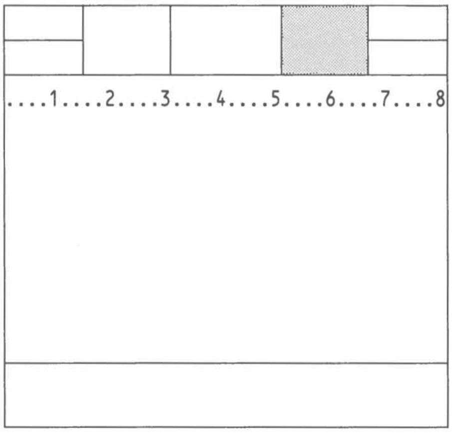


Abbildung 2.9 Textstil

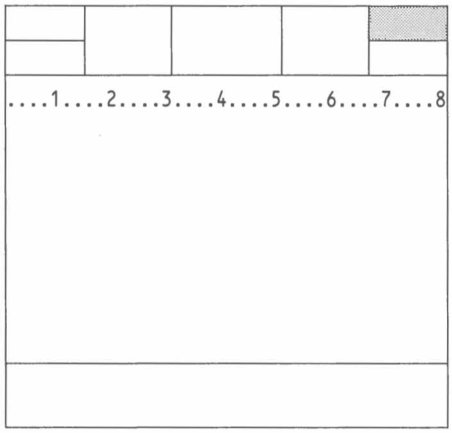


Abbildung 2.10 Die Befehle

BEFEHLE

Die Befehle sind über die Funktionstaste **F3** erreichbar, worauf im Steuerbereich das *Befehlsmenü* angezeigt wird.

Aus diesem Menü können Sie jeden beliebigen Befehl durch Eingabe des betreffenden Anfangsbuchstabens aufrufen.

Dies ist jedoch noch nicht der gesamte Befehlsvorrat: Aus Platzgründen wurden die Befehle auf zwei Menüs verteilt, zwischen denen Sie mit dem Befehl **Weiter** hin– und herschalten können.

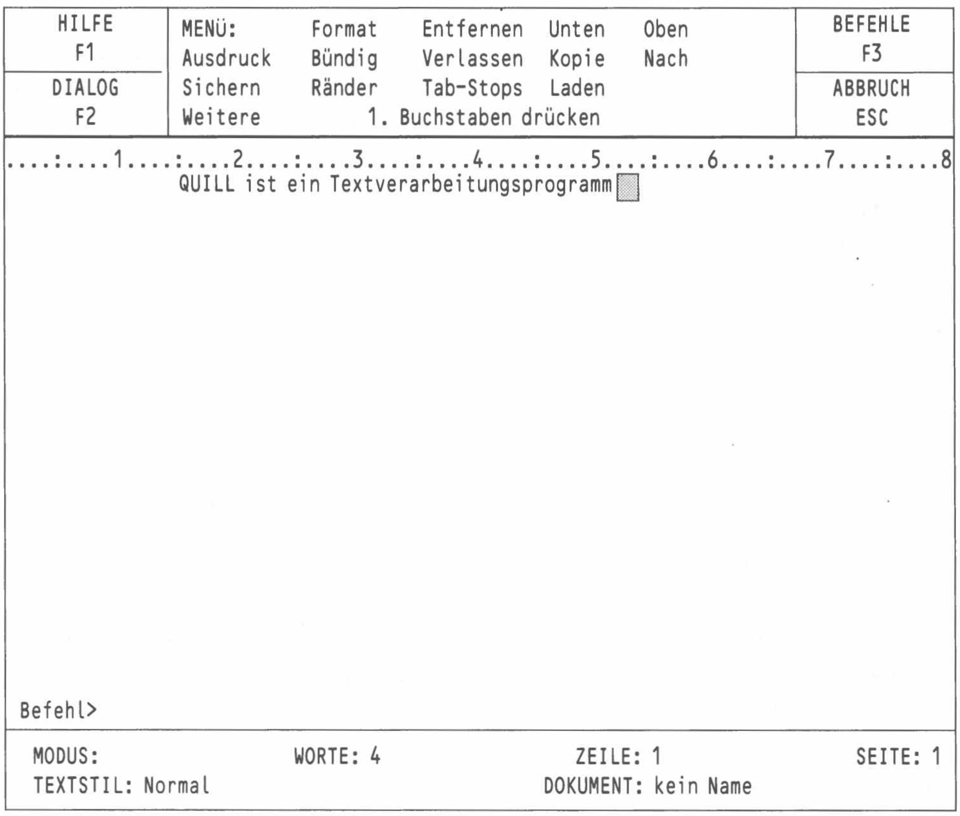


Abbildung 2.11 Das Befehlsmenü

Achten Sie beim Aufrufen der Befehle darauf, daß Sie das richtige Menü vor sich haben, damit der eingegebene Buchstabe auch wirklich den beabsichtigten Befehl startet.

Der verbleibende Teil des Quill-Handbuchs befaßt sich im wesentlichen mit der Beschreibung der einzelnen Befehle. Doch für den Moment beschränken wir uns auf zwei: **Verlassen** und **Tilgen**.

Verlassen dient zum Beenden einer Arbeit mit Quill. Drücken von **F3** und Eingabe von **V** bewirkt den Umstieg von Quill auf SuperBASIC. Der Befehl **Verlassen** fragt an, ob das aktuelle Dokument auf Microdrive-Kassette zu speichern sei. Zur Bejahung **ENTER** drücken, andernfalls das Dokument mit **A** aufgeben.

Mit **ESC** kann der Befehl gegebenenfalls noch annulliert und ins Dokument zurückgekehrt werden.

Für den **Tilgen**-Befehl muß das Menü II aufgerufen werden – also **W** für **Weiter** drücken und erst dann **T** für **Tilgen**; sonst rufen Sie irrtümlich **Tab-Stops** auf. **Tilgen** löscht das aktuelle Dokument aus dem Arbeitsspeicher, ohne jedoch auf SuperBASIC umzusteigen.

```
  HILFE    | MENÜ II:     Dateien   Ersetzen   Tilgen      | BEFEHLE
   F1      | Bindestrich  Mischen   NeueSeite  Suchen      |   F3
  DIALOG   |                                              | ABBRUCH
   F2      | Weitere      1. Buchstaben drücken           |   ESC
....:....1....:....2....:....3....:....4....:....5....:....6....:....7....:....8
                QUILL ist ein Textverarbeitungsprogramm█







 Befehl II>
 MODUS:                WORTE: 4              ZEILE: 1               SEITE: 1
 TEXTSTIL: Normal                            DOKUMENT: kein Name
```

Abbildung 2.12 Das Befehlsmenü II

Wenn Sie einen Text löschen, ohne ihn vorher auf einer Microdrive-Kassette gespeichert zu haben, ist er unwiederbringlich verloren. Aus diesem Grund geht Quill hier auf Nummer Sicher und wartet, bis Sie den Befehl durch Drücken von **ENTER** bestätigen. Falls Sie es sich nochmals anders überlegen, stornieren Sie den Befehl mit **ESC**, der Sie wieder in das Dokument zurückbringt.

KAPITEL 3
CURSOR-EDITIEREN

Dieses Kapitel macht Sie vertraut mit den einfachen Editierfunktionen von Quill. Da Textänderungen nur an der Cursorposition vorgenommen werden können, muß der Cursor mit den Pfeiltasten an die betreffende Stelle gebracht werden.

Man bezeichnet diese Art der Textbearbeitung als *Cursor–Editieren*. Geben Sie einen kurzen Text ein und probieren Sie am besten selbst die verschiedenen Funktionen aus. Achten Sie unbedingt darauf, daß er auch ein paar Tippfehler enthält, die Sie dann übungshalber korrigieren können.

TEXT EINFÜGEN

Der Standardmodus von Quill ist der *Einfügemodus*, wobei der über die Tastatur eingegebene Text automatisch an der Cursorposition eingefügt wird. Buchstaben und Wörter können auch in bereits bestehenden Text eingeschoben werden:

> Den Cursor mit Hilfe der vier Cursorsteuertasten an die Position bringen, wo der Einschub erfolgen soll. Den gewünschten Text eintippen. Die Zeichen werden direkt an der Cursorposition eingegeben, wobei sich der bestehende Text öffnet, d.h. nach rechts verschiebt und Platz für den Einschub macht.

Auch hierbei wird der Text automatisch wieder auf Blockformat justiert.

Bei längeren Einschüben wäre es etwas entnervend, nach jedem Tastenanschlag eine erneute Textformatierung abzuwarten. Quill reagiert auf diese spezielle Situation mit einem Zeilenabbruch an der Stelle, wo der Einschub vorgenommen wird. Man nennt das einen *automatischen Textschnitt*. Danach können Sie beliebig viel Text eingeben.

Sobald Sie Ihren Einschub durch Drücken einer Cursorsteuertaste, einer Funktionstaste oder der **ESC**-Taste abschließen, hängt Quill die auseinandergeschnittenen Textteile wieder aneinander.

TEXT LÖSCHEN

Auch das Löschen von Text an der Cursorposition ist denkbar einfach. Sie benutzen dazu die Cursorsteuertasten in Kombination mit **CTRL**.

Am besten probieren Sie die Wirkung von **CTRL** zusammen mit der Linkspfeiltaste gleich einmal aus. Plazieren Sie den Cursor auf die Stelle direkt nach der Textstelle, die Sie löschen wollen. Halten Sie jetzt **CTRL** fest, während Sie kurz einmal die Linkspfeiltaste anschlagen. Sofort verschwindet das Zeichen links vom Cursor und dieser rückt eine Stelle nach. Durch mehrmaliges Drücken der Linkspfeiltaste löschen Sie mehrere Zeichen nacheinander. Wenn eine ganze Reihe von Zeichen gelöscht werden soll, kann durch längeres Festhalten von **CTRL** und Linkspfeil die Wiederholfunktion in Gang gesetzt werden. Immer zuerst **CTRL** und dann die Cursorsteuertaste drücken.

CTRL zusammen mit dem Rechtspfeil bewirkt einen Löschvorgang, bei dem der Cursor stationär bleibt, während das dort befindliche Zeichen gelöscht wird und das nächste Zeichen mit dem Rest der Zeile die Lücke schließt.

Statt zeichenweise kann das Löschen auch wortweise erfolgen, und zwar sowohl nach rechts wie nach links. Zu diesem Zweck wird zusätzlich zur **CTRL**– noch die **SHIFT**-Taste betätigt.

Bei Bedarf kann auch die Zeile vom Zeilenanfang bis Cursorposition bzw. von der Cursorposition bis zum Zeilenende auf einen Schlag gelöscht werden. **CTRL** in Kombination mit dem Aufwärtspfeil löscht die ganze Zeile links vom Cursor, **CTRL** mit dem Abwärtspfeil löscht den Rest der Zeile.

In allen Fällen nimmt Quill eine automatische Blockformatierung vor.

ÜBERSCHREIBEN

Im Überschreiben-Modus wird alter Text einfach durch neuen überschrieben und ersetzt.

Der Modus-Wechsel erfolgt durch Festhalten von **SHIFT** und Drücken von **F4**. Der Modus-Vermerk im Statusbereich ändert sich dabei von EINFÜGEN auf ÜBERSCHR. Die Rückkehr auf den Einfüge-Modus geschieht durch erneutes Drücken von **SHIFT** und **F4**.

Beim Überschreiben wird der Cursor an die betreffende Stelle gebracht und dann der neue Text darübergetippt. Anschließend daran denken, wieder auf den Einfüge-Modus zurückzuschalten, weil Sie sonst irrtümlich Text zum Verschwinden bringen, der nicht dafür vorgesehen war.

HILFE F1	CURSOR ↑	TEXT Neuer Absatz: ↵	TEXTSTIL	BEFEHLE F3
DIALOG F2	← → ↓	Löschen: CTRL & ←↑↓→ Modus ändern: SHIFT + F4	F4	ABBRUCH ESC

```
....:....1....:....2....:....3....:....4....:....5....:....6....:....7....:....8
                Das ist ein Sazt zum Korrrigieren.
```

MODUS: ÜBERSCHR.	WORTE: 6	ZEILE: 1	SEITE: 1
TEXTSTIL: Normal		DOKUMENT: kein Name	

Abbildung 3.1 Überschreiben

Abbildung 3.1 zeigt eine typische Fehlerkonstellation, bei der sich Überschreiben anbietet. Der Cursor sitzt auf dem "z" von "Sazt", und die Korrektur kann im Überschreiben-Modus durch Eingabe von "tz" erfolgen.

Und wenn Sie schon am Verbessern sind: Löschen Sie auch gleich noch das überflüssige dritte "r" in "Korrrigieren".

KAPITEL 4
DAS TEXTFORMAT

Dieses Kapitel befaßt sich mit dem *Textformat*, d.h. mit dem Layout, der Gliederung und dem optischen Eindruck Ihres Dokuments im Gegensatz zum Textinhalt. In diesem Zusammenhang geht es beispielsweise um verschiedene Schriftarten, wie Fettdruck, Unterstreichung, Hoch- und Tiefstellung, aber auch um die rechte und linke Randeinstellung, um Einrückungen und um den Randausgleich.

TEXTSTIL

Einen speziellen Textstil haben wir bereits kurz erwähnt: die Unterstreichung. An dieser Stelle wollen wir uns ein wenig ausführlicher damit beschäftigen, sowie mit Fettschrift, Hoch- und Tiefstellung.

```
HILFE    | TEXTSTIL Zur Textstil-Änderung Taste        | BEFEHLE
F1       | Fett, Unterstreichen, Tief oder Hoch        | F3
DIALOG   | drücken. Zum Übermalen bzw.                 | ABBRUCH
F2       | Ändern bestehenden Texts Ü drücken.         | ESC
....:....1....:....2....:....3....:....4....:....5....:....6....:....7....:....8
                QUILL ist ein Textverarbeitungsprogramm, das auf hohe
           Leistung, maximale Flexibilität und einfache Bedienung
           ausgelegt ist.
                Sie können eine Textverarbeitungsanlage in jeder
           Situation verwenden, in der Sie normalerweise eine
           Schreibmaschine benutzen würden. Die beiden Maschinen sind
           einander in ihrer Funktionsweise sehr ähnlich. Ein
           automatisches Textverarbeitungssystem bietet jedoch
           wesentliche Vorteile.

  Textstil>
  MODUS:                WORTE: 48            ZEILE: 9              SEITE: 1
  TEXTSTIL: Normal                        DOKUMENT: kein Name
```

4.1 Textstil wählen

Die Wahl dieser Optionen geschieht durch Drücken von **F4** und dem entsprechenden Anfangsbuchstaben – **F** für Fett, **U** für Unterstreichen, **T** für Tief- und **H** für Hochstellen. Ausgeschaltet werden sie auf die gleiche Weise, d.h. durch nochmaliges Drücken von **F4** und dem jeweiligen Anfangsbuchstaben.

Beachten Sie, daß eingegebener Text stets die Schriftart annimmt, die in der TEXTSTIL-Rubrik angezeigt wird. Wenn Sie den Cursor in einen durch Fettschrift hervorgehobenen Textteil bewegen, ändert sich der Stilvermerk auf "Fett", und etwaige Texteingabe erfolgt ebenfalls in dieser Schriftart. Der Textstil ist also an den entsprechenden Textteil gebunden.

Es ist durchaus möglich, gleichzeitig mehrere Funktionen einzuschalten, z.B. Fettdruck und Unterstreichung. Selbstverständlich schließen sich Hoch- und Tiefstellung gegenseitig aus, d.h. die Wahl von **H** schaltet gegebenenfalls **T** aus, und umgekehrt.

Bei der Textstil-Option kann man drei verschiedene Anwendungssituationen unterscheiden:

Eingabe von neuem Text in einer speziellen Schrift
Ändern von bestehendem Text auf eine andere Schriftart
Austauschen oder Entfernen einer ursprünglich gewählten Schrift.

Zur Eingabe von Text in einer bestimmten Schriftart **F4** drücken und die gewünschte Schriftart wählen. Nachfolgender Text erscheint dann solange in diesem Stil, bis die Wahl durch erneutes Drücken von **F4** und dem (den) betreffenden Anfangsbuchstaben wieder annulliert wird.

Die Schriftart eines bereits bestehenden Textes zu ändern, ist ganz einfach. Man spricht dabei von *Übermalen* und stellt sich den Cursor als einen Pinsel vor, der die Schriftart des darunterliegenden Textes ändert.

Als erstes plazieren Sie den Cursor auf den Anfang der Textstelle, drücken **F4** und danach **Ü**. Bestimmen Sie dann die gewünschte(n) Schriftart(en). Steuern Sie den Cursor mit Hilfe der Rechts- und Abwärtspfeiltasten bis ans Ende des betreffenden Textes und verlassen Sie die Option mit **ENTER**. Das Ausschalten der Schriftarten erübrigt sich in diesem Fall; Quill kehrt automatisch in die richtige Schriftart zurück, sobald der überpinselte Bereich verlassen wird. Abbildung 4.2 zeigt die Bildschirmanzeige beim nachträglichen Unterstreichen mit dem Cursor-Pinsel.

```
HILFE    | TEXTSTIL Zur Texstil-Änderung Taste       | BEFEHLE
F1       | Fett, Unterstreichen, Tief oder Hoch      | F3
DIALOG   | drücken, dann bestehenden Text            | ABBRUCH
F2       | mit ↓→ übermalen. ↵ für Ende              | ESC
....:....1....:....2....:....3....:....4....:....5....:....6....:....7....:....8
          QUILL ist ein Textverarbeitungsprogramm, das auf hohe
      Leistung, maximale Flexibilität und einfache Bedienung
      ausgelegt ist.
          Sie können eine Textverarbeitungsanlage in jeder
      Situation verwenden, in der Sie normalerweise eine
      Schreibmaschine benutzen würden. Die beiden Maschinen sind
      einander in ihrer Funktionsweise sehr ähnlich. Ein
      automatisches Textverarbeitungssystem bietet jedoch
      wesentliche Vorteile.

 Übermalen
 MODUS:              WORTE: 48           ZEILE: 1              SEITE: 1
 TEXTSTIL: Unterstr.                     DOKUMENT: kein Name
```

Abbildung 4.2 Nachträgliches Unterstreichen

Umgekehrt können Sie eine ursprünglich gewählte Schriftart nachträglich durch eine andere ersetzen oder einfach entfernen, indem Sie den Cursor an den Anfang der Textstelle setzen und **F4**, gefolgt von **Ü** drücken. Geben Sie dann die entsprechenden Buchstaben zum Ein- bzw. Ausschalten der Option(en) ein. Überpinseln Sie mit dem Cursor die ganze Textstelle und schließen Sie den Vorgang mit **ENTER** ab.

Der ursprüngliche Zustand einer einmal geänderten Schriftart wird von Quill nicht erinnert. Angenommen, eine anfänglich unterstrichene Textstelle wird auf Fettdruck umgestellt und nachträglich die Option Fett wieder entfernt, dann ist das Ergebnis ein Text in Normalschrift (ohne Unterstreichung).

RÄNDER

Ränder dient zum Setzen der Randeinstellungen, welche jeweils vom aktuellen Textabsatz an solange gelten, bis eine neuerliche Änderung eintritt.

Zum Starten des Befehls **F3**, gefolgt von **R** drücken. Darauf erscheinen im Steuerbereich u.a. die drei Optionen **L**(inks), **E**(inrücken) und **R**(echts). Die farblich abgehobene Option kann geändert werden (als erstes also der linke Rand), wobei die verschiedenen Optionen entweder mit der Leertaste oder durch Eingabe des Anfangsbuchstabens angewählt werden. Anschließend ist die betreffende Randeinstellung anhand der Pfeiltasten festzulegen.

Nehmen wir an, Sie wollen den linken Rand um drei Zeichen weiter nach rechts verschieben, beginnend mit dem zweiten Absatz Ihres Dokuments.

Bringen Sie den Cursor zuerst an eine beliebige Stelle im zweiten Absatz und drücken Sie nacheinander die Tasten **F3** und **R**.

Die linke Randeinstellung ist die aktivierte Option. Drücken Sie also dreimal die rechte Cursorpfeiltaste. Die neue Randeinstellung tritt sofort in Kraft, was Sie am Bildschirm mitverfolgen können.

Zum Verlassen des Befehls **ENTER** drücken bzw. zur Modifikation der anderen Randeinstellungen die Leertaste und anschließend zum Verschieben die Rechts– und Linkspfeiltasten betätigen. Die Auf– und Abwärtspfeile dienen zur Plazierung des Cursors auf andere Absätze im Dokument, falls weitere Änderungen vorgenommen werden sollen. Abgeschlossen wird der Befehl mit **ENTER**.

Die Randeinstellung **E**(inrücken) markiert den Anfangspunkt für neue Absätze. Beim Bildschirmformat mit 80 Spalten ist dafür standardmäßig die 15. Position vorgesehen.

Das Verhältnis zwischen dem linken Rand und der Absatzeinrückung kann völlig frei gewählt werden. Werden keine Einrückungen gewünscht, ist beide Male dieselbe Spaltenposition zu wählen. Es ist sogar denkbar, eine "negative Einrückung" vorzusehen, so daß diese weiter links als der linke Rand zu liegen kommt. Praktisch ist dies vor allem bei numerierten Aufstellungen, wie das folgende Beispiel zeigt:

```
Eingerückter (temporärer) Rand
|
|      Linker Rand
1)     Das ist der erste Abschnitt, der
       zeigen soll, in welcher Weise
       man Randeinstellungen und Einrückungen
       verwenden kann.
2)     Der "eingerückte" oder "temporäre" Rand
       liegt drei Zeichenpositionen weiter links
       als der linke Rand.
```

In diesem Beispiel setzt das Drücken von **ENTER** den nachfolgenden Text an die Position des temporären Randes. Weitere Texteingabe erfolgt im normalen Bereich zwischen dem linken und rechten Rand. Jedes Drücken von **ENTER** hebt diese normale Einstellung auf und beginnt den Text an der Position der Einrückung.

RANDAUSGLEICH

Mit dem **Bündig**-Befehl bestimmen Sie den Randausgleich (Justierung) Ihres Dokuments. Analog zum **Ränder**-Befehl gelten auch hier alle Änderungen vom aktuellen Absatz an, d.h. vom Absatz, in dem sich der Cursor gerade befindet. Sie bleiben solange in Kraft, bis eine erneute Änderung vorgenommen wird bzw. bis zum Ende des Dokuments. Beim Aufrufen dieses Befehls werden drei Optionen präsentiert: linksbündig, rechtsbündig oder zentriert (mittig).

Die Standardeinstellung **Bündig R(echts)** bewirkt einen Randausgleich, der auch rechts einen glatten Rand erzeugt. Diese in professionellen Anwendungen übliche Textform (vgl. z.B. den vorliegenden Text) nennt man *Blocksatz*. Der Randausgleich wird durch entsprechende Vergrößerung der Wortzwischenräume erzielt. Der professionelle Eindruck wird leider oft geschmälert, wenn in einem Dokument sehr viele Bandwurmwörter vorkommen.

Für *linksbündigen* Text nach Aufrufen des Bündig-Befehls **L** drücken. Linksbündiger Text, sog. *Flattersatz* ist gekennzeichnet durch einen ausgeglichenen linken und einen ungleichmäßigen rechten Rand. Dafür sind die Wortzwischenräume alle gleich lang. Zur Veranschaulichung des Unterschieds dieser beiden Randausgleich-Einstellungen wurde dieser Abschnitt im Gegensatz zum Rest des Buches linksbündig gesetzt. Der folgende Abschnitt illustriert Zentrierung:

Die Option *Mittig* wird durch Drücken von **M** gewählt und bewirkt, daß der eingegebene Text genau in die Mitte zwischen die zwei Ränder gesetzt wird. Das sieht dann so aus wie der Abschnitt, den Sie gerade lesen. Die zentrierte Darstellung wird sehr oft für Überschriften in Berichten, für den Betreff in Briefen oder für Diagrammbeschriftungen gewählt.

Genau wie beim **Ränder**-Befehl können Sie sich auch hier mit den Auf– und Abwärtspfeilen im Dokument umherbewegen und weitere Änderungen an anderen Absätzen vornehmen. Verlassen wird der Befehl mit **ENTER**.

KAPITEL 5
EDITIERBEFEHLE

In diesem Kapitel stellen wir Ihnen weitere Editierfunktionen vor, welche die Textverarbeitung um vieles erleichtern und beschleunigen. Das sind zum einen das Kopieren, Verschieben und Löschen von ganzen Textblöcken und zum andern die leistungsfähigen Such- und Ersetzoperationen. Quill sieht zu diesem Zweck die Editierbefehle **Kopie**, **Entfernen**, **Suchen** und **Ersetzen** vor.

KOPIE

Kopie dient nicht nur zum Kopieren eines Textblocks, sondern auch zum Verschieben von Text an eine andere Stelle im Dokument.

Im Prinzip sind sich diese zwei Operationen auch sehr ähnlich, nur daß beim eigentlichen Kopieren der ursprüngliche Text an Ort und Stelle belassen und ein Duplikat an eine andere Stelle kopiert wird. Damit sparen Sie sich das mehrmalige Abschreiben desselben Textstücks.

Die Verschiebe-Operation holt den Textblock von seinem ursprünglichen Platz an einen neuen, ohne eine Kopie zu erstellen.

Der Quill-Editierbefehl **Kopie** vereinigt diese beiden Funktionen und überläßt Ihnen die Wahl, ob die ursprüngliche Textkopie gelöscht oder beibehalten werden soll.

Nach Aufrufen von **Kopie** mit **F3** und **K** werden Sie aufgefordert, mit dem Cursor den Anfang des betreffenden Textblocks zu markieren und dann **ENTER** zu drücken. Anschließend den Cursor als Endmarkierung setzen. Dabei wird der zum Kopieren vorgesehene Text farbig hinterlegt. Falls dieser zu lang wird, können Sie jederzeit mit den Links- oder Aufwärtspfeilen eine Korrektur vornehmen; dabei dürfen Sie jedoch nicht über die Anfangsmarkierung zurückgehen. Zum Schluß wieder **ENTER** drücken.

Als nächstes werden Sie gefragt, ob der ursprüngliche Text gelöscht oder beibehalten werden soll. Durch Eingabe von **B** wird ein Duplikat erstellt und der Text an der ursprünglichen Stelle belassen; Drücken von **ENTER** löscht den Text an der alten Stelle. Nun muß Quill noch wissen, wohin die Kopie zu plazieren ist. Markieren Sie die Stelle mit dem Cursor und drücken Sie **K**.

Mit **ENTER** verlassen Sie den Befehl und kehren zur Hauptanzeige zurück.

Falls der gleiche Textblock noch an andere Stellen kopiert werden soll, erledigt Quill auch dies in Sekundenschnelle. Markieren Sie einfach die Positionen, wo die Kopie eingeschoben werden soll, mit dem Cursor und drücken Sie die Taste **K**. Dieser Vorgang kann beliebig oft wiederholt werden. Bei solchen Mehrfachkopien verzichtet Quill auf die Frage, ob die ursprüngliche Kopie gelöscht oder beibehalten werden soll. Drücken Sie zum Verlassen des Befehls **ENTER**.

Auch für Kopierprozeduren gilt, wie immer in Quill, daß die laufende Operation mit **ESC** abgebrochen werden kann, wobei jedoch abgeschlossene Operationen nicht betroffen sind. Bereits eingefügte Kopien gehen nicht verloren, wenn Sie zu irgendeinem Zeitpunkt **ESC** drücken.

ENTFERNEN

Dieser Befehl eignet sich zum Löschen längerer Textstellen aus einem Dokument. Aufgerufen wird er mit **F3** und **E**. Wenn es nur darum geht, ein paar Zeichen zu löschen, ist natürlich die Cursor-Editier-Methode (vgl. Kapitel 3) schneller.

Wie beim **Kopie**-Befehl benutzen Sie auch hier den Cursor als Anfangsmarkierung des zu löschenden Textes und drücken **ENTER**. Anschließend setzen Sie den Cursor auf das letzte Zeichen, welches gelöscht werden soll. Auch hier wird der gesamte Text farbig hinterlegt. Wenn Sie sicher sind, daß der hervorgehobene Text gelöscht werden kann, drücken Sie **ENTER**, worauf der Befehl sofort ausgeführt wird.

SUCHEN

Der **Suchen**-Befehl erlaubt Ihnen, ein Dokument ganz oder teilweise nach einem Suchbegriff zu durchforschen. Dieser Suchbegriff kann ein Zeichen, ein Wort oder eine Wortgruppe sein. Mit **Suchen** können Sie z.B. die Häufigkeit gewisser Wörter oder Floskeln in einem Brief oder Bericht überprüfen. Die erste Suche beginnt am Anfang des Dokuments und kann dann von der Cursorposition aus fortgeführt werden.

Der **Suchen**-Befehl gehört zum Menü II, und der Aufruf kommt folglich mit **F3**, **W**, **S** zustande.

Danach fordert Quill im Statusbereich den gewünschten Suchbegriff an. Schließen Sie Ihre Eingabe mit **ENTER** ab, worauf die Suche sofort aufgenommen wird. Sobald Quill die erste Zeichenfolge findet, die dem Suchbegriff entspricht, hält es an und markiert die Anfangsposition mit dem Cursor. Wenn das die Stelle ist, die Sie haben wollen, können Sie den Befehl mit **ENTER** verlassen.

Ebensogut können Sie jedoch die Suche fortsetzen. Statt **ENTER** drücken Sie in diesem Fall **W** für **Weitersuchen**. Auch diesen Vorgang können Sie beliebig oft wiederholen, bis Sie die gewünschte Stelle finden. Dann mit **ENTER** zur Hauptanzeige zurückkehren.

Falls Quill beim Weitersuchen keine weitere Stelle findet, die auf den Suchbegriff paßt, gibt es die Meldung "Nicht gefunden – weiter mit LEERTASTE" aus und bringt Sie so zur Hauptanzeige zurück.

ERSETZEN

Der **Ersetzen**-Befehl ist dem **Suchen**-Befehl sehr nahe verwandt, nur daß er Ihnen darüberhinaus noch die Möglichkeit bietet, gefundene Zeichenketten auch gleich durch andere zu ersetzen. Auch diesen Befehl finden Sie im Menü II. Wählen Sie also **F3**, **W**, **E**.

Als erstes werden Sie aufgefordert, den Suchbegriff einzugeben. Sobald Sie **ENTER** drücken, findet Quill das erste Vorkommen und bittet um Eingabe des Ersatzbegriffs. Schließen Sie auch diese Eingabe mit **ENTER** ab.

Quill fragt jetzt, ob die gefundene Textstelle tatsächlich durch die neue Zeichenkette ersetzt werden soll. Drücken von **E** bewirkt Ersetzen, Drücken von **N** verhindert das Ersetzen der aktuellen Textstelle. In beiden Fällen sucht Quill nach der nächsten Entsprechung, wo es Ihnen wieder die gleichen Alternativen anbietet. Der Vorgang bricht ab, wenn keine weiteren Entsprechungen gefunden werden können bzw. bei Drücken von **ENTER**.

Quill gibt eine Meldung aus, wenn es keine Textstellen mehr findet, die dem Suchbegriff entsprechen, und wartet ab, bis Sie die Leertaste für die Rückkehr zur Hauptanzeige drücken.

Der **Ersetzen**-Befehl hat vielseitige Verwendungsmöglichkeiten, wenn es darum geht, Texte zu überarbeiten. Er kann zum Ersetzen, Hinzufügen und Löschen von Begriffen dienen, die in einem Dokument häufig vorkommen, wie im folgenden Beispiel gezeigt werden soll. (Ein Beispiel aus der Geographie.)

Ersetzen der Bezeichnung "Fluß" durch "Strom": Geben Sie "Fluß" als den Suchbegriff und "Strom" als den Ersatzbegriff ein.

Ergänzen der Bezeichnung "Fluß" durch "Fluß oder Bach": Geben Sie "Fluß" als den Suchbegriff und "Fluß oder Bach" als den Ersatzbegriff ein.

Löschen des Begriffs "und Voralpen" im Ausdruck "Alpen und Voralpen": Geben Sie als Suchbegriff "und Voralpen" ein und drücken Sie auf die Frage "Ersetzen durch" einfach **E**. (Der Ersatzbegriff ist eine sog. Nullkette.)

KAPITEL 6

WEITERE FORMATIER-FUNKTIONEN

In diesem Kapitel wollen wir uns den verbleibenden Optionen zur Formatierung des Textes zuwenden. Dazu gehört das Setzen der Tabulatoren und der Seitenumbrüche und Entscheidungen, die mit dem Erscheinungsbild des Textes zu tun haben, wie etwa der Zeilenabstand oder die Seitenlänge. Diese Optionen bestimmen im wesentlichen das Aussehen Ihres Dokuments.

TABS

Tab-Stops setzen

Tab-Stops sind ein altbekanntes Hilfsmittel zur Gliederung von Text und Tabellen. Es sind speziell markierte Spaltenpositionen im Text. Beim Drücken der **TAB**-Taste springt der Cursor aus seiner momentanen Position nach rechts zum nächsten Tab-Stop in derselben Zeile. Befindet er sich rechts vom letzten Tab-Stop, springt er in die Anfangsposition der folgenden Zeile.

Zur besseren Orientierung zeigt Ihnen Quill direkt unter der Spaltenskala eine Tabulatorskala mit den Tab-Positionen und –Typen an.

Quill sieht verschiedene Tab-Typen vor, die Sie an beliebigen Positionen setzen können. In einer Zeile dürfen bis zu 16 Tab-Stops gesetzt werden.

Tab-Stop-Typen

Quill unterscheidet vier verschiedene Typen von Tab-Stops.

Der gebräuchlichste ist der *Links*-Tab-Stop, der ganz genau so funktioniert wie auf der traditionellen Schreibmaschine. Durch Drücken der **TAB**-Taste springt der Cursor auf die nächste Tab-Position, die den Anfangspunkt für die folgende Texteingabe bildet. Der Name Links-Tab kommt daher, weil der Text spaltenweise linksbündig angeordnet wird.

Nun zum nächsten Typ, dem *Rechts*-Tab-Stop. Wenn Sie sich auf einem solchen befinden und Text eingeben, bleibt der Cursor stationär, während der Text links davon erscheint und bündig mit dem Tab-Stop angeordnet wird. Dies dauert solange, bis der verfügbare Platz links vom Tab gefüllt ist bzw. bis Sie erneut die **TAB**-Taste drücken und zum folgenden Tab-Stop springen. Mit Rechts-Tabs errichtete Zeilen kommen spaltenweise rechtsbündig untereinander zu stehen.

Der dritte Typ ist der zentrierte, *mittige* Tab-Stop. Eingegebener Text wird um den Tab-Stop zentriert, und zwar solange, bis der verfügbare Platz aufgebraucht ist (begrenzt durch bestehenden Text oder durch den linken Rand) bzw. bis Sie erneut **TAB** drücken.

Schließlich kommen wir zum vierten Typ, dem *Dezimal*-Tab, der Ihnen beim Eintippen von Zahlenkolonnen viel Mühe erspart, weil die Dezimalpunkte aller Zahlen automatisch untereinander zu stehen kommen. Falls eine Eingabe keinen Dezimalpunkt enthält, funktioniert dieser Tab wie ein Rechts-Tab.

Abbildung 6.1 illustriert die Wirkung der verschiedenen Tab-Typen auf den Text.

Links	Mittig	Rechts	Dezimal
|	|	|	|
ein	ein	ein	ein
kurzes Stück	kurzes Stück	kurzes Stück	kurzes Stück
Text	Text	Text	Text
|	|	|	|
12.345	12.345	12.345	12.345
123.4	123.4	123.4	123.4
1234.56	1234.56	1234.56	1234.56
|	|	|	|

Abbildung 6.1 Die vier Tab-Stop-Typen

Der Befehl Tab-Stops

Anfänglich ist jede zehnte Zeichenposition ein Links-Tab-Stop. Mit dem Befehl **Tab-Stops** können Sie die Anzahl, die Position und den Typus ändern.

Dabei ist es ganz Ihnen überlassen, wo Sie welche Art von Tab-Stop plazieren, solange Sie nicht mehr als 16 Tabs in eine Zeile setzen. Neue Tab-Stops gelten vom aktuellen Textabsatz an (dem Absatz, in dem sich der Cursor bei Aufruf des Befehls **Tab-Stops** befindet) bis zum Ende des Dokuments bzw. bis zur nächsten Tabs-Änderung.

Die Wahl des Befehls **Tab-Stops** (**F3** und **T**) bewirkt die Anzeige aller Tab-Positionen auf der Tabulatorskala.

Den Typus der Tab-Stops erkennen Sie an den Buchstaben (**L**, **M**, **R**, **D**). Der Cursor sitzt in der Anfangsposition und läßt sich mit den Pfeiltasten hin– und herbewegen.

Sie können beliebig viele Änderungen vornehmen und mit Hilfe der Auf– und Abwärtspfeile zu anderen Textabsätzen übergehen und dort Tab-Änderungen ausführen. Zum Beenden des Befehls **Tab-Stops** mit **ENTER** auf die Hauptanzeige zurückkehren.

Tabs einfügen

Zum Einfügen eines Tabs wählen Sie zunächst den Typus und markieren dann mit dem Cursor die gewünschte Position, indem Sie **T** drücken.

Die vier Typen werden Ihnen jeweils nach Drücken von **F3** und **T** im Steuerbereich angezeigt, nämlich **L**(inks), **M**(ittig), **R**(echts) und **D**(ezimal).

(**L**)inks ist hervorgehoben, d.h. aktiviert. Folglich wird der nächste Tab-Stop ein Links-Tab, es sei denn, Sie wechseln den Typus.

Zum Wechseln können Sie entweder die Leertaste benutzen, die jeweils den nächsten Typus aktiviert, oder den betreffenden Anfangsbuchstaben. Für einen Dezimal-Tab drücken Sie entweder dreimal die Leertaste oder einmal den Buchstaben **D**.

Tabs löschen

Zum Annullieren eines **Tab-Stops** den Cursor direkt auf die betreffende Spaltenposition bringen und dann **X** drücken.

FORMAT

Mit dem **FORMAT**-Befehl bestimmen Sie das Aussehen der Bildschirmmaske, welche z.B. durch Faktoren wie

Zeichen pro Zeile
Zeilenabstand
Seitenlänge (in Zeilen)

beeinflußt wird.

Abbildung 6.2 zeigt die Präsentation des Befehls. Eine ausführliche Beschreibung der einzelnen Optionen finden Sie in Kapitel 8.

Aufgerufen wird der **FORMAT**-Befehl mit **F3** und **F**, worauf Quill das Optionsmenü anzeigt. Angenommen, Sie möchten die Anzeigebreite (das Bildschirmformat) auf 40 Zeichen ändern. Drücken Sie den Anfangsbuchstaben **A**. Sofort wird die entsprechende Auswahlzeile hell hinterlegt, und Quill erwartet die Eingabe der gewählten Anzeigebreite. Sie können erst weitermachen, wenn Sie 4, 6 oder 8 gewählt haben. (Anzeigebreite 40, 64 oder 80 Zeichen pro Zeile.)

HILFE F1	FORMAT regelt die Gestaltung der ausgedruckten Seite	BEFEHLE F3
DIALOG F2	Den 1. Buchstaben der Option drücken Für Ende ↵ drücken	ABBRUCH ESC

```
Unterer Rand (Zeilenzahl) ............................................ 3
Anzeigebreite 80,64,40 (8,6,4) ....................................... 8
Zeilenabstand (0,1,2) ................................................ 0
Seitenlänge (Zeilenzahl) ............................................. 66
Beginn Seite Nr ...................................................... 1
Textfarbe - Grün oder Weiß ........................................... Grün
Oberer Rand (Zeilenzahl) ............................................. 6
```

Abbildung 6.2 Der FORMAT-Befehl

Danach können Sie beliebig andere Auswahlzeilen ändern und den **FORMAT**-Befehl mit **ENTER** wieder verlassen.

BREITE DOKUMENTE

Wenn Sie den rechten Rand so weit verschieben, daß die Anzahl von Zeichen in der Zeile größer ist als die Anzeigebreite Ihres Sichtgeräts, dann ist es nicht möglich, die ganze Breite des Dokuments gleichzeitig zu sehen. Der Arbeitsbereich ist dann wie ein Fenster, welches Ihnen jeweils einen Ausschnitt des Dokuments zeigt, und zwar immer in Abhängigkeit von der Cursorposition. Um sich eine andere "Aussicht" zu verschaffen, verschieben Sie einfach den Cursor.

Seitenlänge

Eine Option im **Format**-Befehl ist die Seitenlänge, ausgedrückt als die maximale Zeilenzahl je Seite. Dabei muß berücksichtigt werden, daß diese Zahl nicht nur die eigentlichen Textzeilen umfaßt, sondern auch den oberen und den unteren Seitenrand, die Kopf- und die Fußzeile und den Abstand, der diese vom Haupttext trennt.

Nehmen wir an, Sie haben einen oberen Seitenrand von 3 Zeilen, eine Kopfzeile mit 2 Zeilen Abstand bis zum Haupttext; danach vier Leerzeilen bis zur Fußzeile und einen unteren Seitenrand von 5 Zeilen. Das macht zusammen bereits 3+1+2+4+5 = 15 Zeilen. Bei einer Seitenlänge von 66 Zeilen bleiben Ihnen 66–15 = 51 Zeilen für Text pro Seite. Wenn Sie dazu auch noch einen doppelten Zeilenabstand wählen – d.h. jeweils eine Leerzeile zwischen den Textzeilen – dann bleiben Ihnen lediglich 26 Zeilen für Text übrig.

Seitenumbruch

Der Seitenumbruch markiert das Ende einer Seite bzw. den Anfang einer neuen. Er bestimmt sich aus der Einstellung der Seitenlänge im **Format**-Befehl und ist gekennzeichnet durch eine waagerechte Linie und eine Seitennummer. Quill berücksichtigt bei der Berechnung der Seitenlänge den oberen und den unteren Rand, die Kopf- und Fußzeilen usw. Wenn wir in unserem vorher erwähnten Beispiel einen Zeilenabstand 0 annehmen, also Text auf jeder Zeile, dann sieht Quill jeweils nach 51 Textzeilen einen Seitenumbruch vor.

Wenn Sie Ihre Seitenlänge so wählen, daß auf eine Seite weniger als 5 Zeilen passen, dann schaltet Quill den automatischen Seitenumbruch aus. In diesem Fall werden keine Seitenumbrüche und keine Zählung angezeigt; Quill behandelt das gesamte Dokument wie eine einzige fortlaufende Seite. Bei Bedarf können Sie also den automatischen Seitenumbruch ausschalten, indem Sie die Seitenlänge auf Null setzen.

Seitenumbruch erzwingen

Wenn ein Seitenumbruch erzwungen werden soll, gibt es den Befehl **Neue Seite**. Nützlich ist dies z.B. zu Anfang eines neuen Kapitels, oder um sicherzustellen, daß eine Tabelle nicht auf zwei Seiten verteilt wird.

Zum Einfügen eines solchen obligatorischen Seitenumbruchs an irgendeiner Stelle in Ihrem Dokument rufen Sie den Befehl **Neue Seite** mit **F3**, **W**, **N** und markieren dann mit dem Cursor die letzte Zeile vor dem gewünschten Seitenumbruch, indem Sie ihn an eine beliebige Position setzen und **N** drücken. Quill geht dann am Ende der Zeile auf eine neue Seite über.

Ihr Dokument kann mehrere erzwungene Seitenumbrüche enthalten, jedoch immer nur einen pro Zeile. Zum Verlassen des Befehls **ENTER** drücken.

Bereits gesetzte Seitenumbrüche können auch wieder gelöscht werden. Dazu wird nicht etwa der Befehl **Neue Seite** verwendet. Stattdessen bringen Sie den Cursor einfach auf den betreffenden Seitenumbruch und halten dann **CTRL** fest, während Sie gleichzeitig den Linkspfeil drücken.

KAPITEL 7
DATEIBEFEHLE

Es ist keine schlechte Idee, ein geschriebenes oder überarbeitetes Quill-Dokument auf Microdrive-Kassette abzuspeichern. Auf diese Weise können Sie es irgendwann später wieder abrufen, neu bearbeiten oder ausdrucken, und außerdem schützen Sie sich gegen Datenverlust.

Ein Dokument, welches auf Microdrive-Kassette gespeichert wird, nennt man eine *Datei*. Dateien sind zusammengehörige Informationsbündel, denen ein eigener Name zugewiesen wird. Quill verfügt über eine Reihe von Befehlen zum Umgang mit Dateien, mit denen wir uns im folgenden beschäftigen wollen.

SICHERN

Dieser Befehl dient zum Speichern eines Dokuments auf einer Microdrive-Kassette. Ein nicht auf diese Weise gesichertes Dokument geht unwiederbringlich verloren, sobald Sie Quill verlassen.

Nach Aufrufen von **Sichern** mit **F3** und **S** werden Sie von Quill nach einem Namen gefragt. Die ganze Sequenz zum Sichern des aktuellen Dokuments hat diese Form:

[F3] `S Brief01` [ENTER]

Damit wird Ihr Dokument unter dem Namen "Brief01_doc" auf der Kassette im Microdrive 2 abgespeichert.

Falls sich dort bereits eine Datei mit dem gleichen Namen befindet, gibt Quill eine Warnung aus und fragt vorsichtshalber an, ob das alte Dokument durch das neue überschrieben werden soll. Drücken Sie zur Bejahung **J** und im Zweifelsfall lieber **ESC**, um das aktuelle Dokument unter einem anderen Namen zu sichern und das alte intakt zu lassen.

Sobald Quill den Speichervorgang erledigt hat, fragt es, ob Sie am aktuellen Dokument weiterarbeiten möchten, was mit **ENTER** zu bestätigen ist. Andernfalls **ESC** drücken, um ein anderes Dokument in Angriff zu nehmen.

Beim Benennen eines Dokuments zum Sichern bzw. beim Laden eines bereits auf Kassette befindlichen Dokuments zeigt Quill im Statusbereich den Dateinamen an. Wenn Sie zu einem späteren Zeitpunkt die bearbeitete Version abspeichern wollen, schlägt Quill den aktuellen Namen des Dokuments als Dateinamen vor. Wenn Sie damit einverstanden sind, drücken Sie **ENTER**. Die Sequenz zum Sichern des aktuellen Dokuments ist in diesem Fall ganz einfach:

[F3] `S` [ENTER]

worauf die neueste Version des Dokuments die früher abgespeicherte überschreibt. Genausogut können Sie aber die alte Version aufbewahren und der neuen einen anderen Namen zuweisen.

LADEN

Mit dem **Laden**-Befehl holen Sie sich eine auf Kassette gespeicherte Datei in den Arbeitsspeicher und machen sie zum aktuellen Dokument, welches dann bearbeitet und geändert werden kann.

Als erstes werden Sie nach dem Namen des gewünschten Dokuments gefragt. Falls Sie sich nicht mehr erinnern, geben Sie ein Fragezeichen ein und **ENTER**, worauf Quill Ihnen mit einem Katalog aller Dateien im Microdrive 2 zu Hilfe kommt. Geben Sie dann den Namen ein.

Die Eingabe eines falschen Namens quittiert Quill mit der Meldung, daß eine solche Datei nicht existiert. Probieren Sie es nochmals.

DATEIEN UND MISCHEN

Der **Dateien**-Befehl bietet die vier Optionen:

Reserve	kopiert ein Microdrive-Dokument oder eine -datei
Löschen	löscht ein Microdrive-Dokument oder eine -datei
Formatieren	formatiert eine Microdrive-Kassette
Import	holt eine aus Abacus oder Archive stammende Microdrive-Datei an die Cursorposition im aktuellen Dokument.

Mit dem **Mischen**-Befehl können Sie ein Dokument von Microdrive-Kassette in das aktuelle Dokument hereinholen und es an der Cursorposition einschieben.

Im Zusammenhang mit den soeben beschriebenen Dateibefehlen werden Sie oft mit zwei Datenkassetten gleichzeitig arbeiten, z.B. um eine Reservekopie auf einer anderen Kassette zu speichern oder um eine Datei zu importieren, die auf einer anderen Kassette gesichert wurde.

Zu diesem Zweck können Sie die Quill-Kassette aus dem Microdrive 1 herausnehmen und dort eine andere Kassette einschieben. Vergessen Sie nur nicht, die Quill-Kassette wieder einzulegen, falls Sie ein Dokument ausdrucken oder die Hilfe-Datei aufrufen wollen. Wenn Ihr System mit zusätzlichen Microdrives ausgerüstet ist, brauchen Sie normalerweise die Quill-Kassette in Microdrive 1 nicht zu entfernen.

AUSDRUCK

Mit diesem Befehl können Sie sich ein Dokument ganz oder teilweise auf Papier ausdrucken lassen. Selbstverständlich geht das nur, wenn Ihrem System ein betriebsfähiger Drucker angeschlossen ist. Ohne Drucker nützt der Befehl nicht viel.

Quill geht davon aus, daß Sie das aktuelle Dokument ausdrucken wollen und wartet Ihre Bestätigung ab: **ENTER** druckt das aktuelle Dokument; soll ein anderes ausgedruckt werden, geben Sie den entsprechenden Namen ein. Voraussetzung ist, daß sich das genannte Dokument im Microdrive 2 befindet.

Quill bietet an, das ganze Dokument zu drucken. Bestätigen Sie dies mit **ENTER** oder geben Sie die Seitenzahlen ein, wo der Ausdruck beginnen und enden soll. Nach jeder Seitenzahl **ENTER** drücken. Es werden immer ganze Seiten gedruckt.

Der Befehl **Ausdrucken** kann das Dokument auch an eine Microdrive-Datei senden statt an den Drucker. Mit **ENTER** wird der Drucker als Bestimmungsort für die Ausgabe gewählt, mit einem Dateinamen die entsprechende Datei. Eine auf diese Weise erzeugte Datei enthält alle Zeichen, einschließlich der Steuerzeichen, die andernfalls für den Drucker bestimmt gewesen wären.

Die einfachste Form des Befehls ist das Ausdrucken des ganzen aktuellen Dokuments. Die Eingabe lautet dann:

[F3] A [ENTER] [ENTER] [ENTER]

Soll nur ein Teil des Dokuments gedruckt werden, z.B. die Seiten 2 bis 4 des Dokuments mit dem Namen "Brief01_doc" auf eine neue Datei namens "Brief01_lis" (beide im Microdrive 2), ist folgende Eingabe vorzunehmen:

[F3] A Brief01 [ENTER] 2 [ENTER] 4 [ENTER] Brief01 [ENTER]

Bevor Quill den Druckauftrag ausführt, liest es den Druckertreiber von der Quill-Kassette in Microdrive 1, um sich über die verfügbaren Druckerfunktionen und deren Steuerung zu informieren. Quill kann mit den meisten Druckern arbeiten. Für eine ausführlichere Beschreibung zum Betrieb von Druckern verweisen wir auf die Beschreibung des Druckertreiberprogramms im *Informationsteil.*

Etwaige Änderungen am Druckformat, z.B. Zeilenabstand und Seitenlänge, sind unter Zuhilfenahme des **Format**-Befehls vorzunehmen, der in Kapitel 6 beschrieben ist.

KAPITEL 8
QL QUILL ÜBERSICHT

DIE FUNKTIONSTASTEN

Neben den Standardfunktionen **F1**, **F2** und **F3** erfüllt die Funktionstaste 4 die folgenden Aufgaben:

F4	Ändern von Textstil/Schriftart
SHIFT F4	Umschalten zwischen Einfügen und Überschreiben

(F5 wird von Quill nicht benutzt)

Zur Wahl eines Befehls **F3** drücken. Quill präsentiert das erste Befehlsmenü. Den Cursor können Sie nach wie vor umherbewegen, aber jegliche Veränderung des Textes (Eingabe oder Löschen) ist gesperrt.

Der Steuerbereich zeigt jetzt die Liste der verfügbaren Befehle, welche durch Eingabe ihres Anfangsbuchstabens aufgerufen werden können. Eine Reihe weiterer Befehle befindet sich auf dem MENÜ II. Zwischen diesen beiden Menüs wird mit **W** für Weitere hin- und hergeschaltet.

Aufgepaßt: Einige Anfangsbuchstaben können sich sowohl auf einen Befehl im ersten wie auch im zweiten Menü beziehen. Aus diesem Grund ist beim Befehlsaufruf immer darauf zu achten, daß der gewünschte Befehl auch angezeigt ist.

Allgemein gilt, daß jede in Ausführung begriffene Operation mit **ESC** abgebrochen werden kann.

Nach Beendigung der meisten Befehle kehrt Quill zur Hauptanzeige zurück. Ausnahmen bilden nur jene Befehle, die eigene, befehlsinterne Menüs haben (z.B. **Dateien**), aus denen die Rückkehr zur Hauptanzeige mit **ESC** zu erfolgen hat.

Bei Befehlen, die eine Eingabe vom Benutzer anfordern (z.B. **Sichern**, **Laden**, **Dateien**, **Ersetzen**) werden etwaige Tippfehler oder fehlerhafte Angaben mit dem Zeileneditor korrigiert, der in der Einführung zu den QL-Programmen beschrieben ist.

DIE BEFEHLE

Es folgt eine alphabetische Zusammenstellung aller in Quill verfügbaren Befehle. Befehle aus MENÜ II wurden entsprechend gekennzeichnet (II).

AUSDRUCKEN

Mit diesem Befehl kann das im Arbeitsspeicher befindliche Dokument bzw. eine auf der Kassette im Microdrive 2 befindliche Datei ganz oder teilweise ausgedruckt werden.

Zum Ausdrucken des aktuellen Dokuments **ENTER** drücken, andernfalls den gewünschten Dateinamen und **ENTER** eingeben.

Quill bietet an, das gesamte Dokument auszudrucken. Bestätigen Sie dies mit **ENTER** oder spezifizieren Sie gegebenenfalls die auszudruckenden Seiten durch Angabe der ersten Seitenzahl (gefolgt von **ENTER**) und der letzten Seitenzahl (gefolgt von **ENTER**).

Schließlich schicken Sie den Text mit **ENTER** an den Drucker bzw. durch Eingabe des Namens einer Zieldatei auf eine Microdrive-Datei.

Vor Ausführung des Druckauftrags liest Quill die Datei "printer_dat" mit den notwendigen Druckertreiber-Informationen.

BINDESTRICH (II)

Mit diesem Befehl ist es möglich, Positionen für Worttrennungen zu bestimmen. Kommt die vorgesehene Trennung zustande, weil das Wort einen Zeilenumbruch bewirkt, wird automatisch ein Trennstrich eingefügt. Wörter, die nicht in dieser Weise gekennzeichnet sind, werden niemals getrennt, sondern immer als ganze auf die folgende Zeile gesetzt.

Bei Verwendung von Blocksatz (Randausgleich) empfiehlt es sich, besonders bei sehr langen Wörtern, solche Trennstriche vorzusehen, weil auf diese Weise die Entstehung unschöner Wortzwischenräume verhindert wird und ein harmonischeres Erscheinungsbild erzielt werden kann.

Den Cursor auf den Silbenanfang setzen, d.h. auf den Buchstaben, der abgetrennt werden soll, und **B** drücken. Sie können beliebig viele Trennvorschläge in dieser Weise vornehmen. Zum Verlassen des Befehls **ENTER** drücken.

Bei Wörtern, die keinen Zeilenumbruch verursachen, bleibt der Bindestrich-Befehl ohne sichtbare Wirkung.

BÜNDIG

Dieser Befehl bestimmt die Anordnung des Textes im Verhältnis zu den beiden Randeinstellungen. Er gilt vom aktuellen Textabsatz bis zum Ende des Dokuments bzw. bis zur nächsten Modifikation.

Zur Wahl stehen die folgenden drei Optionen, die mit der Leertaste oder mit den jeweiligen Anfangsbuchstaben zu aktivieren sind:

Links	Linksbündige Textanordnung mit Flatterrand rechts
Mittig	Zentrierte Textanordnung
Rechts	Blocksatz mit beidseitigem Randausgleich. Der glatte rechte Rand kommt durch Vergrößerung der einzelnen Wortzwischenräume zustande.

Um Änderungen an anderen Absätzen vorzunehmen, den Cursor mit Hilfe der Auf- und Abwärtspfeile steuern und wie beschrieben vorgehen. Zum Schluß **ENTER** drücken.

DATEIEN (II)

Vier Optionen stehen hier zur Wahl:

Löschen
: Löscht ein benanntes Dokument bzw. eine Datei von einer Microdrive-Kassette. Sie werden nach dem Namen der Datei gefragt, die gelöscht werden soll. Drücken von **ENTER** bewirkt sofortige Ausführung des Befehls.

Formatieren
: Formatiert die Kassette in Microdrive 2. Da durch das Formatieren alle bereits auf der Kassette befindlichen Informationen verlorengehen, wartet Quill sicherheitshalber Ihre Bestätigung ab.

  Vorsicht: Der Formatiervorgang löscht alle Daten auf der Kassette unwiederbringlich.

Reserve
: Erstellt eine Zweitkopie eines Dokuments auf einer Microdrive-Kassette. Empfiehlt sich, um wertvolle Daten zu sichern. Sie geben den Dokumentennamen und den Namen der Reservekopie ein. Im allgemeinen ist es am besten, die Reservekopie auf einer anderen Kassette zu speichern, wobei derselbe Name gewählt werden kann.

Import
: Fügt eine andere Datei von einer Microdrive-Kassette in das aktuelle Dokument ein. Der Einschub erfolgt an der Cursorposition. Die importierte Datei muß eine aus QL Abacus oder QL Archive bzw. eine Textdatei (z.B. aus SuperBASIC) sein. Näheres im Informationsteil.

ENTFERNEN

Dieser Befehl dient zum Löschen längerer Textstellen. Sie markieren als erstes den Anfang des zu löschenden Blocks und drücken **ENTER**. Anschließend bringen Sie den Cursor auf das letzte Zeichen, das gelöscht werden soll. Dabei wird der zum Löschen vorgesehene Text farbig hinterlegt. Sobald Sie **ENTER** drücken, wird die Löschprozedur ausgeführt.

ERSETZEN (II)

Dieser Befehl ermöglicht es Ihnen, eine Zeichenkette (Zeichen, Wort, Wortgruppe) selektiv durch eine andere zu ersetzen.

Geben Sie den Suchbegriff ein, gefolgt von **ENTER**, und anschließend den Ersatzbegriff, gefolgt von **ENTER**.

Quill durchsucht das Dokument von Anfang an, bis es auf das erstmalige Vorkommen des Suchbegriffs stößt. Jetzt haben Sie die Wahl, den Suchbegriff gegen den Ersatzbegriff auszutauschen. Drücken von **E** veranlaßt Quill, den Ersetzungsvorgang durchzuführen, Drücken von **N** beläßt den alten Begriff an dieser Stelle.

Quill führt die Suche weiter bis zur nächsten Zeichenkette, die dem Suchbegriff entspricht und wartet wiederum auf Ihre Entscheidung. Diese Prozedur wiederholt sich bis ans Ende des Dokuments bzw. bis zum Beenden des Befehls mit **ENTER**.

FORMAT

Mit diesem Befehl haben Sie die Möglichkeit, das endgültige Aussehen Ihres Dokuments zu beeinflussen. Die verfügbaren Optionen, die Ihnen beim Aufrufen des Befehls vorgelegt werden, sind:

Unterer Rand
Geben Sie die Anzahl der Leerzeilen am Seitenende an und drücken Sie **ENTER**. Die Standardvorgabe ist 3.

Anzeigebreite
Drücken Sie 4, 6 oder 8 zur Einstellung von 40, 64 oder 80 Zeichen pro Zeile. Quill akzeptiert keine andere Eingabe. Die Standardeinstellung ist 80 bzw. 64, je nachdem, ob Sie mit einem Monitor oder einem Fernseher arbeiten.

Zeilenabstand
Wählen Sie 0, 1 oder 2. Die Zahlen entsprechen den Leerzeilen, die zwischen die einzelnen Textzeilen eingeschoben werden. Andere Eingaben werden nicht akzeptiert. Standardeinstellung ist 0.

Seitenlänge
Geben Sie die Gesamtzeilenzahl pro Seite ein und drücken Sie **ENTER**. Diese Zahl schließt auch die für den oberen und unteren Rand vorgesehenen Zeilen und die Kopf- und Fußzeilen ein. Durch Eingabe von 0 verhindern Sie den automatischen Seitenumbruch. Standardmäßig ist eine Seitenlänge von 66 Zeilen vorgesehen.

Beginn Seite Nr.
Geben Sie die gewünschte Seitennummer ein, mit der die Paginierung beginnen soll, und **ENTER**. Alle folgenden Seiten werden fortlaufend numeriert. Der Standardwert ist 1, doch kann er auf eine beliebige Zahl gesetzt werden, wenn das Dokument z.B. eine Fortsetzung eines anderen ist und eine kontinuierliche Zählung gewünscht wird.

Textfarbe
Zur Wahl der Farben für Normal- und Fettschrift. Diese Option weist diesen beiden Schriftarten alternativ grün und weiß zu. Anfänglich wird Normalschrift grün, Fettschrift weiß dargestellt.

Oberer Rand
Geben Sie die Anzahl von Leerzeilen für den oberen Seitenrand ein, gefolgt von **ENTER**. Standardmäßig werden 6 Zeilen vorgesehen.

Nach jeder Option können Sie entweder zu einer weiteren übergehen oder den Befehl mit **ENTER** verlassen.

KOPIE

Dieser Befehl dient zum Kopieren und zum Verschieben von Textblöcken innerhalb eines Dokuments.

Als erstes werden Sie aufgefordert, mit dem Cursor den Anfang des zu kopierenden Textes zu markieren und **ENTER** zu drücken. Anschließend kennzeichnen Sie das Ende des Textblocks in der gleichen Weise. Der Cursor kann auch zurückverschoben werden, jedoch niemals über die Anfangsmarkierung hinaus. Der zum Kopieren vorgesehene Text wird farbig hinterlegt. Wenn Sie soweit sind, **ENTER** drücken. Auf die Anfrage mit **ENTER** bestätigen, daß der Originaltext gelöscht werden kann, oder ihn mit **B** beibehalten. Danach mit dem Cursor die Stelle kennzeichnen, wo die Kopie einzuschieben ist, und **K** drücken.

Weitere Kopien können an beliebigen anderen Stellen im Dokument eingefügt werden, indem Sie den Cursor dorthin bringen und **K** drücken. Den Befehl mit **ENTER** verlassen.

LADEN

Dieser Befehl holt ein Dokument von einer Microdrive-Kassette in den Arbeitsspeicher, damit es editiert oder ausgedruckt werden kann.

Geben Sie den Namen des gewünschten Dokuments ein, d.h. den Dateinamen, den Sie ihm beim Sichern zugewiesen hatten. Eingabe von **?** und **ENTER** bewirkt die Anzeige des Dateienkatalogs im Microdrive 2. Wird ein anderer Microdrive gewünscht, kann der Systemvorschlag entsprechend geändert werden. Quill fordert nach Ausgabe der Liste nochmals den Dateinamen an.

MISCHEN (II)

Mit diesem Befehl wird eine Kopie eines benannten, auf Microdrive-Kassette abgespeicherten Dokuments erstellt und an der Cursorposition im aktuellen Dokument eingefügt.

Bei Verwendung von **Mischen** kann gegebenenfalls die Quill-Kassette in Microdrive 1 herausgenommen und eine andere Datenkassette eingelegt werden. Allerdings müssen Sie nach Verlassen des Befehls wiederum die Quill-Kassette in den Microdrive 1 schieben.

Plazieren Sie den Cursor vor Aufrufen des Befehls an die Stelle, wo der Einschub erfolgen soll. Quill fordert dann den Namen der Datei an, die eingefügt werden soll.

Sie können die Datei auch mitten in einem bestehenden Textabsatz einfügen; dieser wird dann an der Cursorposition automatisch von Quill aufgeteilt und das Dokument dazwischen eingeschoben.

NACH

Mit Hilfe dieses Befehls bringen Sie den Cursor ganz schnell an den Textanfang, das Textende oder auf eine bestimmte Seite im Dokument. Die drei Sprungoptionen sind:

Anfang	Cursor auf Textanfang
Ende	Cursor ans Textende
Seiten-Nr.	Eingabe einer Seitenzahl, gefolgt von **ENTER**, versetzt den Cursor sofort auf die betreffende Seite im Dokument. Bei einem nicht-paginierten Dokument (ohne Seitenumbruch) springt er ans Textende.

NEUE SEITE (II)

Verwenden Sie diesen Befehl, um den Übergang auf eine neue Seite zu markieren – einen sog. obligatorischen Seitenumbruch.

Den Cursor auf die Zeile setzen, an deren Ende der Seitenumbruch stattfinden soll, und **N** drücken.

Sie können mit dieser Methode beliebig viele Seitenumbrüche in ein Dokument einfügen (nicht mehr als einen pro Zeile). Zum Verlassen des Befehls **ENTER** drücken.

Annulliert werden Seitenumbrüche *nicht* mit **Neue Seite**. Setzen Sie den Cursor von unten kommend an eine beliebige Stelle innerhalb der betreffenden Seitentrennlinie. Drücken Sie **CTRL** zusammen mit dem **Linkspfeil**.

OBEN

Mit diesem Befehl können Sie eine sog. Kopfzeile definieren, die im gedruckten Dokument jeweils am oberen Seitenrand erscheint. Auf dem Bildschirm werden Kopfzeilen nicht angezeigt. Kopfzeilen müssen ausdrücklich mit dem **Oben**-Befehl definiert werden, da Quill automatisch keine vorsieht.

Folgende Varianten stehen zur Auswahl:

Kein Text	Dokument ohne Kopfzeile (Standardvorgabe)
Links	Linksbündiger Kopfzeilentext
Mittig	Zentrierter Kopfzeilentext
Rechts	Rechtsbündiger Kopfzeilentext

Aktivieren Sie die gewünschte Option mit der **Leertaste** und **ENTER**. Danach fordert Quill den Kopfzeilentext an. Diese Eingabe mit **ENTER** abschließen.

Wurde bereits früher im Dokument eine Kopfzeile definiert, wird diese im Statusbereich angezeigt und für Änderungen freigegeben, wodurch Sie sich das Neueintippen sparen.

Die Kopfzeile kann auch eine Seitennummer enthalten, deren Position und Typ Sie mit Hilfe eines aus drei Buchstaben bestehenden Schlüssels festlegen:

Schlüssel	Paginierungstyp
`nnn` oder **`NNN`**	arabische Ziffern, d.h. 1, 2, 3, 4
`rrr` oder **`RRR`**	römische Ziffern, d.h. I, II, III, IV
`aaa` oder **`AAA`**	alphabetisch, d.h. A, B, C, D

RÄNDER

Dieser Befehl dient zum Setzen und Ändern der Randeinstellungen Ihres Dokuments – linker Rand, rechter Rand und Einrückung. Die vorgenommenen Änderungen können Sie am Bildschirm mitverfolgen.

Im Steuerbereich werden die Optionen **L**inks, **E**inrücken und **R**echts angezeigt, wobei anfänglich immer Links aktiviert, d.h. zur Modifikation mit Hilfe der Cursorsteuertasten, freigegeben ist.

Mit der Leertaste können Sie auf die anderen Optionen umstellen. Zum Setzen der Randeinstellungen wird der Cursor mit dem Links– und Rechtspfeil auf die betreffende Spalte plaziert.

Die neuen Randeinstellungen gelten von dem Absatz, in dem der Cursor bei Eingabe des Befehls steht, und bleiben solange gültig, bis eine neuerliche Änderung vorgenommen wird, bzw. bis zum Ende des Dokuments.

Selbstverständlich können Sie die Ränder an mehreren Textabsätzen verändern. Bewegen Sie den Cursor entsprechend mit den Auf– und Abwärtspfeiltasten. Zum Verlassen des Befehls **ENTER** drücken.

SICHERN

Dieser Befehl dient zum Speichern eines Dokuments auf eine Microdrive-Kassette.

Zur Identifikation muß jedes Dokument mit einem Namen versehen werden, unter dem es dann in Form einer Datei abgespeichert wird. Wird kein Name eingegeben, sondern einfach **ENTER** gedrückt, weist Quill dem Dokument wieder den ursprünglichen Namen zu. Beim Speichern wird dann die alte Version überschrieben.

Anschließend fragt Quill, ob dasselbe Dokument weiter bearbeitet werden soll. **ENTER** bewirkt, daß das Dokument im Arbeitsspeicher verfügbar bleibt.

Um mit einem anderen Dokument fortzufahren, drücken Sie die Leertaste.

SUCHEN (II)

Dieser Befehl durchsucht Ihr Dokument nach dem spezifizierten Suchbegriff. Dies kann ein Zeichen, ein Wort oder eine Wortgruppe sein.

Geben Sie als erstes den Suchbegriff ein. Sobald Sie **ENTER** drücken, beginnt Quill den Text von Anfang an zu durchsuchen, bis es auf die erste Textstelle stößt, die der gesuchten Zeichenkette entspricht.

Mit **W** veranlassen Sie Quill zur Fortsetzung der Suche. Wenn Sie die gewünschte Textstelle gefunden haben, können Sie den Befehl mit **ENTER** verlassen.

TAB-STOPS

Mit dem **Tab-Stops**-Befehl bestimmen Sie die Positionen und die Art der **Tab-Stops**. Der Sprung von einem Tab zum nächsten erfolgt mit der TAB-Taste. Gesetzte **Tab-Stops** gelten vom aktuellen Absatz (d.h. vom Absatz, in dem der Cursor sich gerade befindet) bis ans Ende des Dokuments bzw. bis zur nächsten Tab-Änderung.

Quill kennt vier verschiedene Tab-Stop-Typen:

Links	Der Links-Tab funktioniert wie ein linker Rand: der Text wird spaltenweise linksbündig angeordnet.
Mittig	Der zentrierte Tab bildet den Mittelpunkt, um den der Text zentriert wird.
Rechts	Der Rechts-Tab funktioniert wie ein rechter Rand: der Text wird spaltenweise rechtsbündig angeordnet.
Dezimal	Der Dezimal-Tab-Stop markiert die Stelle für den Dezimalpunkt bei Zahlen bzw. funktioniert wie ein Rechts-Tab.

Die **Tab-Stops** erkennen Sie auf dem Bildschirm auf der Tabulatorskala, wobei die verschiedenen Tab-Typen wie folgt definiert sind:

L – Links-Tab-Stop
M – Mittiger, zentrierter Tab-Stop
R – Rechts-Tab-Stop
D – Dezimal-Tab-Stop

Der Cursor sitzt in der ersten Spalte der Tabulator-Zeile. Gesteuert wird er mit dem Links– und dem Rechtspfeil.

Zum Löschen eines **Tab-Stops** bewegen Sie den Cursor auf die betreffende Stelle und drücken **X**.

Zum Einfügen eines **Tab-Stops** bestimmen Sie erst den Typus, den Sie entweder mit Hilfe der Leertaste oder durch Eingabe des Anfangsbuchstabens (**L**, **M**, **R** oder **D**) aktivieren. Markieren Sie dann die gewünschte Tab-Position mit dem Cursor und drücken Sie **T**.

Tabs können beliebig gelöscht und eingefügt werden, und Änderungen können an verschiedenen Textabsätzen vorgenommen werden, indem Sie den Cursor mit dem Auf– und Abwärtspfeil dorthin steuern. Wenn alle Tabulatoren gesetzt sind, verlassen Sie den Befehl mit **ENTER** und kehren zur Hauptanzeige zurück.

TILGEN (II)

Vorsicht: Dieser Befehl löscht das ganze aktuelle Dokument, ohne es auf Microdrive-Kassette abzuspeichern. Der Arbeitsbereich wird geleert, und Sie können mit einem neuen Dokument arbeiten.

UNTEN

Mit diesem Befehl können Sie eine sog. Fußzeile einrichten, einen kurzen Text oder eine Zahl, die am Fußende jeder Seite erscheint – allerdings nur auf dem Papierausdruck, nicht am Bildschirm.

Als erstes wählen Sie die Plazierung aus den vier Optionen:

Kein Text	Wenn keine Fußzeile gewünscht wird
Links	Linksbündige Plazierung
Mittig	Zentrierter Fußtext (Standardvorgabe)
Rechts	Rechtsbündige Plazierung

Mit der Leertaste aktivieren Sie die gewünschte Option und drücken dann **ENTER**. Danach fordert Quill die Eingabe des für die Fußzeile vorgesehenen Textes. Den Text mit **ENTER** abschließen.

Falls bereits ein Fußzeilentext besteht, wird Ihnen dieser im Statusbereich zur Änderung vorgelegt, um Ihnen etwas Tipparbeit zu sparen.

Die Fußzeile kann auch eine Seitennummer enthalten, deren Position und Stil Sie mit Hilfe eines aus drei Buchstaben bestehenden Schlüssels festlegen:

Schlüssel	Paginierungstyp
nnn oder NNN	arabische Ziffern, d.h. 1, 2, 3, 4
rrr oder RRR	römische Ziffern, d.h. I, II, III, IV
aaa oder AAA	alphabetisch, d.h. A, B, C, D

Schließlich werden Sie noch nach einer Zahl zwischen 0 und 9 gefragt, welche den Abstand zwischen dem Haupttext und der Fußzeile festlegt.

VERLASSEN

Befehl zum Verlassen von Quill und Übergang auf SuperBASIC. Drei Optionen werden angeboten:

ENTER	Zum Abspeichern (Sichern) des aktuellen Dokuments vor Verlassen von Quill. Dem Dokument kann auf Wunsch ein neuer Name zugewiesen werden; ansonsten drücken Sie einfach **ENTER**, womit die überarbeitete Version mit dem gleichen Namen auf Kassette abgespeichert wird und die alte Version überschreibt.
A	Aufgeben des aktuellen Dokuments, wenn Sie sich entscheiden, die zuletzt bearbeitete Version nicht zu speichern und zu SuperBASIC zurückzukehren.
ESC	Befehl rückgängig machen und ins aktuelle Dokument zurückkehren.

WEITERE

Dieser Befehl ermöglicht das Hin– und Herwechseln zwischen den beiden Befehlsmenüs. Die Listen der Befehle werden im Statusbereich angezeigt, der sich jeweils beim Drücken von **W** ändert.

Da einige Anfangsbuchstaben in beiden Menüs vorkommen und also zwei verschiedene Befehle bezeichnen können, müssen Sie vor Aufrufen des Befehls sicherstellen, daß er tatsächlich in der aktuellen Liste enthalten ist.

TEXTSTIL

Dieser Befehl gestattet Ihnen gewisse Änderungen am Schriftbild Ihres Dokuments. Mit der Funktionstaste **F4** holen Sie sich die vier unten beschriebenen Stil-Optionen auf den Bildschirm, die Sie durch Eingabe des betreffenden Anfangsbuchstabens wählen. Die gewählte Schriftart gilt bis zur nächsten Stil-Modifikation.

Eine weitere Möglichkeit ist das sog. Übermalen von bereits bestehendem Text mit einer anderen Schriftart. Diese Funktion rufen Sie mit **F4** und **Ü** auf.

Die folgenden Schriftarten stehen zur Auswahl:

Fettdruck	zur Hervorhebung von Text in fetter Schrift
Hochstellung	Text eine halbe Zeile über die Grundzeile
Tiefstellung	Text eine halbe Zeile unter die Grundzeile
Unterstreichung	zur Hervorhebung von Textstellen oder Überschriften

Diese drucktechnischen Optionen können in beliebiger Kombination gewählt werden, mit Ausnahme der beiden Funktionen Hoch– und Tiefstellung, die sich natürlich gegenseitig ausschließen, so daß die Wahl der einen die andere automatisch annulliert.

Um mehrere Schriftarten gleichzeitig zu aktivieren: **F4**, gefolgt von den entsprechenden Buchstaben drücken.

Nach Wahl von **Ü** zum Übermalen wählen Sie ebenfalls den gewünschten Textstil. Markieren Sie den Anfang der Textstelle, die übermalt werden soll, mit dem Cursor und drücken Sie **ENTER** zum Beenden der Funktion. Quill schaltet danach automatisch auf den vorausgegangenen Textstil zurück.

Das Ausschalten funktioniert genau wie das Einschalten: **F4** und danach den betreffenden Anfangsbuchstaben (**F**, **U**, **T** oder **H**) drücken.

EINFÜGEN UND ÜBERSCHREIBEN

Der Standardeingabemodus von Quill ist der Einfüge-Modus. Dabei wird neuer Text an der Cursorposition in das Dokument eingefügt. Bereits bestehender Text rückt nach rechts und macht Platz.

Auf den zweiten Modus, Überschreiben, schalten Sie durch Festhalten von **SHIFT** und Drücken von **F4**. Hier wird alter Text einfach überschrieben, d.h. es wird jeweils das Zeichen an der Cursorposition durch ein neues ersetzt.

Mit **SHIFT** und **F4** schalten Sie wieder zurück auf den Einfüge-Modus.

DIE ANFANGS-PARAMETER

Direkt nach dem Laden befindet sich Quill im Anfangszustand, wobei die unten aufgeführten Standardparameter gelten. Nach Bedarf können diese mit der rechts angegebenen Methode geändert werden.

Parameter	Standard		Ändern mit
Modus	Einfügen		**SHIFT** + **F4**
Anzeigebreite	80 (Mon)	64 (TV)	**Format**
Linksrand	10	0	**Ränder**
Einrückungsspalte	15	5	**Ränder**
Rechtsrand	70	64	**Ränder** (max. 160)
Oberer Seitenrand	6		**Format**
Unterer Seitenrand	3		**Format**
Textjustierung	rechts		**Bündig**
Tab-Stops	links, Spalten 10,20,...80		**Tab-Stops**
Seitenlänge	66		**Format**
Zeilenabstand	0		**Format**
Kopfleiste	keine		**Oben**
Fußleiste	zentriert, "Seite nnn"		**Unten**
Erste Seitenzahl	1		**Format**
Textfarbe			
Normal	grün		**Format**
Fett	weiß		**Format**
Textstil/Schriftart:			
Fett	aus		F4
Unterstreichung	aus		F4
Hochstellung	aus		F4
Tiefstellung	aus		F4

KAPITEL 1

WAS IST QL QUILL 1

KAPITEL 2

FANGEN SIE AN 2
QL QUILL LADEN 2
ANZEIGEFORMAT 2
Der Anzeigenbereich 3
Der Statusbereich 3
Der Steuerbereich 4
DER CURSOR 4
TEXT 4
TEXTSTIL 5
BEFEHLE 6

KAPITEL 3

CURSOR-EDITIEREN 8
TEXT EINFÜGEN 8
TEXT LÖSCHEN 8
ÜBERSCHREIBEN 8

KAPITEL 4

DAS TEXTFORMAT 10
TEXTSTIL 10
RÄNDER 11
RANDAUSGLEICH 12

KAPITEL 5

EDITIERBEFEHLE 13
KOPIE 13
ENTFERNEN 13
SUCHEN 13
ERSETZEN 14

KAPITEL 6

WEITERE FORMATIER-FUNKTIONEN 15
TABS 15
Tabs-Stops setzen 15
Tab-Stop-Typen 15
Der Befehl Tab-Stops 15
Tabs einfügen 16
Tabs löschen 16
FORMAT 16
BREITE DOKUMENTE 17
Seitenlänge 17
Seitenumbruch 17
Seitenumbruch erzwingen 17

KAPITEL 7

DATEIBEFEHLE 18
SICHERN 18
LADEN 18
DATEIEN UND MISCHEN 18
AUSDRUCK 19

KAPITEL 8

QL QUILL ÜBERSICHT 20
DIE FUNKTIONSTASTEN 20
DIE BEFEHLE 20
AUSDRUCKEN 20
BINDESTRICH (II) 20
BÜNDIG 21
DATEIEN (II) 21
ENTFERNEN 21
ERSETZEN (II) 21
FORMAT 21
KOPIE 22
LADEN 22
MISCHEN (II) 22
NACH 23
NEUE SEITE (II) 23
OBEN 23
RÄNDER 23
SICHERN 24
SUCHEN (II) 24
TAB-STOPS 24
TILGEN (II) 25
UNTEN 25
VERLASSEN 25
WEITERE 25
TEXTSTIL 26
EINFÜGEN UND ÜBERSCHREIBEN 26
DIE ANFANGS-PARAMETER 26

sinclair

QL

QL Abacus

WICHTIG

BEVOR SIE DIE APPLIKATIONEN BENUTZEN, LESEN SIE BITTE DIE NACHSTEHENDE BESCHREIBUNG, UM ARBEITSKOPIEN ZU ERSTELLEN.

ARBEITSKOPIEN

Fertigen Sie bitte immer mindestens eine Arbeitskopie an, bevor Sie eines der vier QL Programme benutzen. Arbeiten Sie dann immer nur mit der Arbeitskopie. Bewahren Sie die Originalkassetten mit den Programmen sicher auf. Benutzen Sie die Originalkassetten nur, um bei Bedarf Kopien anzufertigen. So schützen Sie sich vor dem Verlust Ihrer Programme.

Bei allen magnetischen Speichermedien, also auch bei Microdrive-Kassetten, kann bei Gebrauch Datenverlust entstehen. Daher sollten Sie alle Programme und Daten, die Sie noch benötigen, mindestens zweimal speichern. Fertigen Sie sich deshalb immer Sicherungskopien an. So schützen Sie sich vor Datenverlust. Immer, wenn Sie einer Kassette Daten oder Programme hinzufügen, sollten Sie das auch bei der Sicherungskopie tun. Wenn Sie das versäumen, laufen Sie Gefahr, alles, was Sie seit der letzten Datensicherung hinzugefügt haben, zu verlieren. Hinweise zur Anfertigung von Sicherungskopien finden Sie im Abschnitt *Information*.

by Dick de Grandis-Harrison *(Psion Limited)*

KAPITEL 1
WAS IST QL ABACUS?

QL Abacus ist ein Tabellenkalkulationsprogramm. Es dient zum Erfassen von Daten, zum Aufstellen von Budgets und Planungsunterlagen sowie zur Berechnung, Speicherung und Präsentation von Zahlenmaterial. QL Abacus ist ein *Arbeitsblatt* in Tabellenform, eingeteilt in 255 horizontale Zeilen und 64 Spalten. Die Bildschirmmaske zeigt jeweils nur einen Ausschnitt der ganzen Tabelle, ist also eine Art *Fenster*, das Ihnen den Blick auf einen Teilbereich eröffnet. Das Fenster läßt sich seitlich und auf- und abwärts verschieben.

Die Schnittpunkte der Spalten und Zeilen bilden mehr als 16000 *Felder* oder Zellen. Diese Felder sind für die Eingabe von Text und zur Aufnahme von Zahlen und Daten vorgesehen.

Die eigentliche Stärke von Abacus liegt jedoch in der Verwendung von Berechnungsvorschriften oder *Formeln*, welche verschiedene Blöcke, Zeilen oder Spalten, oder auch einzelne Felder rechnerisch miteinander verknüpfen. Dies bedeutet, daß Informationen aus einem Tabellenbereich ausgewertet und sofort in veränderter Form an einer anderen Stelle übernommen werden können.

Zum Beispiel ist es denkbar, zwölf Spalten als Monatskolonnen einzurichten und eine Zeile für die Verkaufsziffern. Eine weitere Zeile enthält eine Formel zur Berechnung der Kosten (z .B. 15% des Verkaufspreises plus feste Kosten), und die Zeile darunter errechnet den Gewinn. Auf diese Weise wird bei jedem Eintrag einer Verkaufzahl automatisch der Gewinn kalkuliert, den Sie in der untersten Zeile sofort ablesen können. Mit Hilfe einer weiteren Formel werden die Monatsbeträge zu einem Jahrestotal aufaddiert. Jede Änderung eines Wertes, etwa der Verkaufszahlen im Monat März, wirkt sich auf die Gewinnkurve und das Jahrestotal aus. QL Abacus rechnet alle Werte automatisch neu durch und zeigt die Ergebnisse sofort an.

Ein großer Vorzug ist die Möglichkeit, Abacus-Daten in andere Psion QL-Programme zu *exportieren*, sie beispielsweise bildlich in Form von Tortendiagrammen darzustellen oder in tabellarischer Form in ein Quill-Dokument einzuflechten.

In vieler Hinsicht ist Abacus eine visuelle Programmiersprache, die mühelos zu handhaben ist. Sie gestattet auch den Umgang mit Text, Daten oder Formeln und kann Eingabe- und Ausgabeanweisungen sowie Textvariablen bewältigen.

Wenn Sie zu irgendeinem Zeitpunkt nicht sicher sind, wie es weitergeht, können Sie mit **F1** die Hilfe-Datei aufrufen. Außerdem sei daran erinnert, daß noch nicht abgeschlossene Operationen (Eingabe von Werten, Starten eines Befehls) mit **ESC** rückgängig gemacht werden können.

KAPITEL 2
FANGEN SIE AN

QL ABACUS LADEN

Laden Sie QL Abacus gemäß der Anleitung in der Einführung zu den QL-Programmen. Nach abgeschlossener Ladeprozedur meldet sich das System mit dieser Anzeige:

```
LADEN DER QL ABACUS
Kalkulation
Version x.y
Copyright ©1984 PSION Ltd.
Alle Rechte vorbehalten
```

wobei x.y die Versions-Nummer bedeutet, etwa 2.23.

Vor dem Start legt das Programm eine kurze Pause ein.

Die Hilfe-Datei wird nicht zusammen mit dem eigentlichen Abacus–Programm in den Arbeitsspeicher geladen, sondern erst bei Bedarf von der Kassette eingelesen. **Aus diesem Grund sollten Sie die Abacus-Originalkassette im Microdrive 1 belassen, wenn Sie vorhaben, zu irgendeinem Zeitpunkt die Hilfsfunktion aufzurufen.**

Die anfängliche Bildschirmmaske von Abacus ist die *Hauptanzeige*, wie sie in Abbildung 2.1 dargestellt ist.

ANZEIGEFORMAT

Abacus verfügt über drei Anzeigeformate: 80, 64 oder 40 Spalten (Zeichenpositionen) pro Zeile. Bei Verwendung eines handelsüblichen Fernsehers ist das Bild unter Umständen nicht scharf genug für das 80er Format, und die Anzeigebreite mit 64 (bzw. 40) Spalten ist vermutlich besser lesbar. Der Bildschirmaufbau bei 80 und bei 64 Spalten ist sehr ähnlich; etwas anders ist der 40-spaltige, wie Sie beim Vergleich der beiden Abbildungen 2.1 und 2.2 leicht sehen.

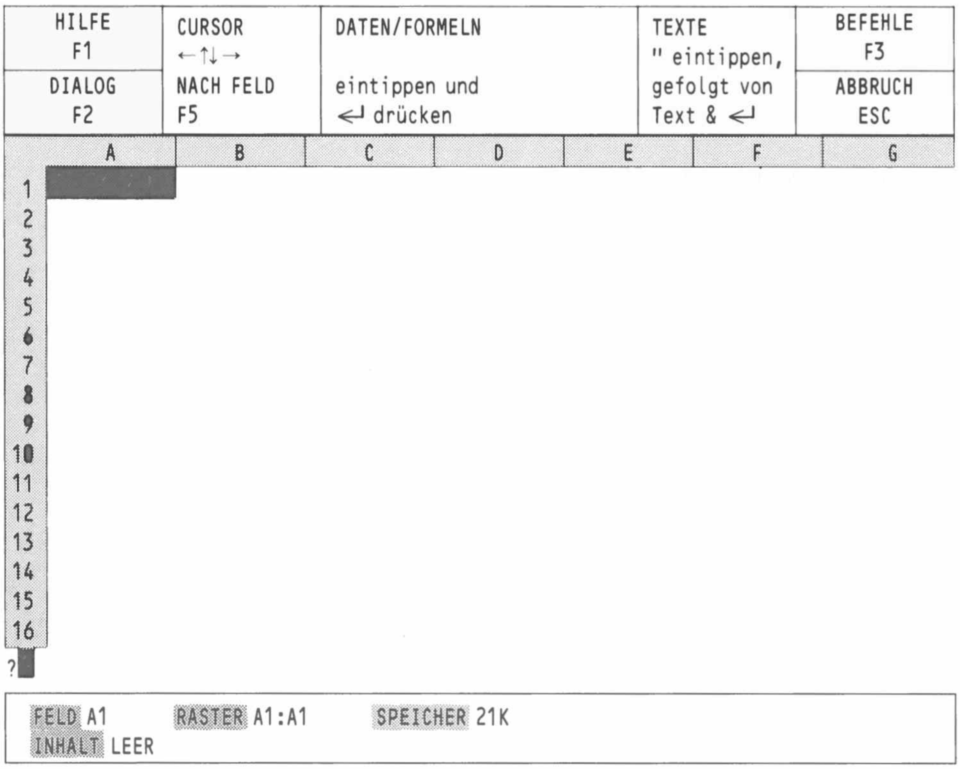


Abbildung 2.1 Die Hauptanzeige bei einem Monitor (80 Spalten pro Zeile)

Je nachdem, ob Sie beim Start **F1** oder **F2** drücken, schaltet Abacus automatisch auf das Format mit 80 bzw. mit 64 Spalten.

Abgesehen vom optischen Unterschied im Bildschirmaufbau funktioniert Abacus in allen drei Anzeigearten genau gleich. Bei den Abbildungen wurde meist das 80er Format verwendet.

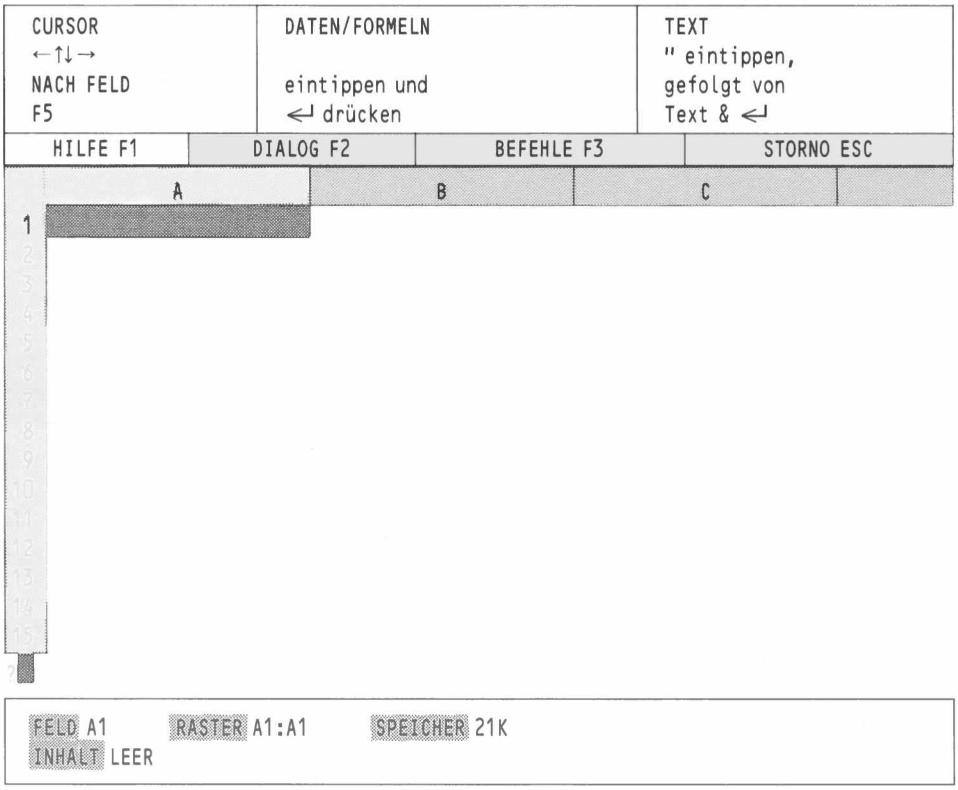


Abbildung 2.2 Die Hauptanzeige bei 40-spaltiger Anzeigebreite

Das Fenster

Der mittlere Bereich der Bildschirmmaske ist das *Fenster*, welches einen Ausschnitt der Tabelle (des Rasters) zeigt.

Eingerahmt wird die Tabelle oben durch einen horizontalen Streifen, der Buchstaben (A, B, C...) enthält. Jeder Buchstabe markiert eine vertikale Spalte. Am linken Rand wird die Tabelle durch einen vertikalen Streifen mit Zahlen (1, 2, 3 bis 16) abgeschlossen. Diese Zahlen bezeichnen die waagerechten Zeilen des Rasters.

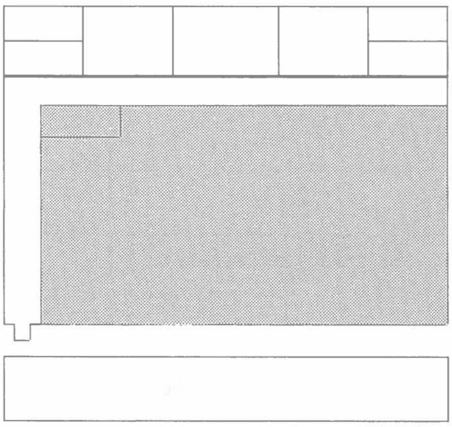


Abbildung 2.3 Das Fenster

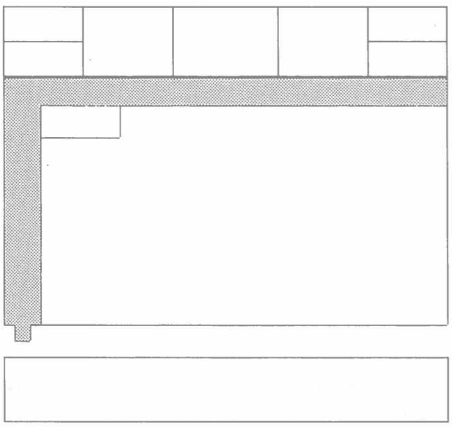
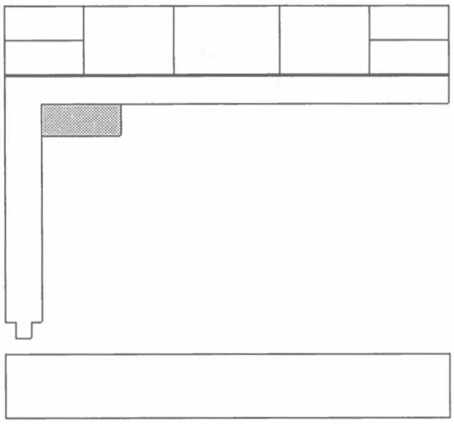


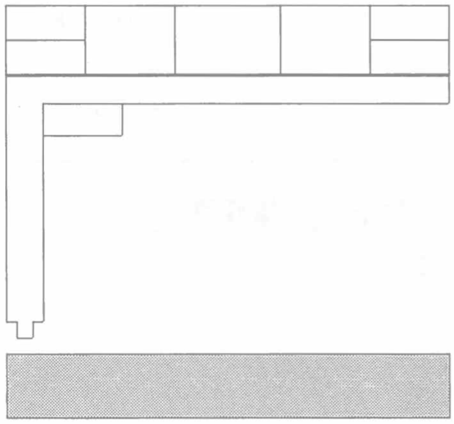
Abbildung 2.4 Die Spalten- und Zeilenbezeichnungen

Jedes Feld befindet sich im Schnittpunkt einer Spalte und einer Zeile, die man sich als Koordinaten vorstellen kann. Von daher kann jedes Feld durch Angabe des betreffenden Spalten-Buchstabens und der Zeilen-Zahl eindeutig identifiziert werden. Man nennt das eine *Feldadresse* oder einen *Feldverweis*. So bezeichnet die Feldadresse A1 z.B. das Feld in Spalte A, Zeile 1, d.h. das erste Feld in der oberen linken Fensterecke.

Dieses Feld wird im Moment von einem roten länglichen Block eingenommen, dem *Cursor*, der das *aktuelle Eingabefeld* markiert.



2.5 Der Cursor



2.6 Der Statusbereich

Der Statusbereich

Der unterste Bereich in der Bildschirmmaske ist der *Statusbereich*, dem Sie Informationen über den momentanen Zustand der Tabelle entnehmen.

So finden Sie dort die Feldadresse des aktuellen Feldes und seinen Inhalt – beim Starten von Abacus ist das Feld leer. Außerdem sehen Sie einen Vermerk über den bisher belegten Tabellenbereich (Adresse der äußersten Felder links oben und rechts unten, die besetzt sind) und die verfügbare Speicherkapazität.

Der Steuerbereich

Der Steuerbereich informiert über die allgemeinen Optionen für Hilfe (**F1**), zum Ein– und Ausblenden der Dialogmeldungen (**F2**), zum Aufrufen des Befehlsmenüs (**F3**) und zum Abbrechen einer Operation (**ESC**). Die drei Felder in der Mitte enthalten drei Abacus-spezifische Optionen, nämlich:

Cursorsteuerung
Eingabe von Daten oder Formeln
Texteingabe

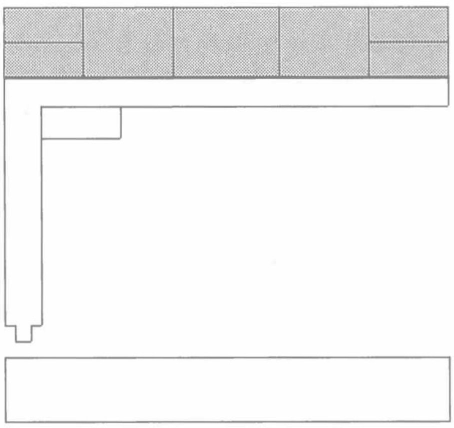


Abbildung 2.7 Der Steuerbereich

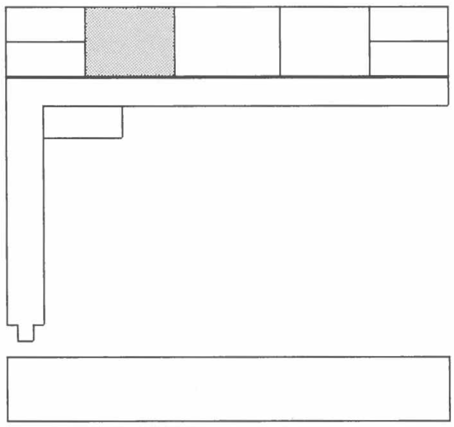


Abbildung 2.8 Cursorsteuerung

CURSOR-STEUERUNG

Der Cursor wird mit den vier Pfeiltasten in der Tabelle umherbewegt. Ein kurzer Tastendruck auf den Rechtspfeil bewirkt einen Sprung in die benachbarte Spalte nach rechts. Wenn Sie einen Blick auf den Feldanzeiger im Statusbereich werfen, sehen Sie dort die Adresse B1. Durch Drücken der Linkspfeiltaste bringen Sie den Cursor wieder auf seinen Ausgangsplatz. Ein weiterer Linkssprung ist nicht möglich, da Sie sich am linken Rand der Kalkulationstabelle befinden.

Bewegen Sie jetzt den Cursor an den extrem rechten Rand des Fensters. Wenn Sie in dieser Position weiter die Rechtspfeiltaste betätigen, können Sie keine Cursorverschiebung mehr beobachten, sondern eine ständige Veränderung der Spaltenbezeichnungen im obersten Streifen des Fensters. Anders ausgedrückt: ein Versuch, den Cursor über den rechten Rand des Bildschirms hinauszubewegen, bewirkt eine Verschiebung des Fensters. Dabei bleibt der Cursor weiterhin sichtbar.

Die Cursorsteuertasten sind eine recht bequeme Fortbewegungsmöglichkeit, wenn nur einige Felder zu überspringen sind. Für weite Distanzen gibt es jedoch eine schnellere Methode, mit der direkt in ein bestimmtes Zielfeld gesprungen wird. Durch Drücken von **F5** rufen Sie die Option **Nach** auf, welche eine Feldadresse anfordert. Schließen Sie Ihre Eingabe mit **ENTER** ab.

Hier eine kleine Übung für einen Sprung auf das Zielfeld D11. **F5** drücken, worauf die Anfrage **Nach > A1** direkt unterhalb der letzten Zeile des Fensters erscheint. Abacus schlägt vor, auf Feld A1 zu springen. Wenn Ihnen das genehm ist, drücken Sie **ENTER**, andernfalls geben Sie eine andere Feldadresse ein, in unserem Fall

d11

gefolgt von **ENTER**. Die Schreibung ist unerheblich, Abacus akzeptiert Groß- und Kleinbuchstaben. Ihre eingegebene Feldadresse überschreibt die ursprüngliche, und der Cursor springt sofort in das Zielfeld.

Bewegen Sie jetzt den Cursor wieder in seine Ausgangsposition, Feld A1, indem Sie dieselbe Methode benutzen und diesmal die Standardvorgabe akzeptieren. Sie brauchen also nur folgende Eingabe vorzunehmen:

[F5] [ENTER]

Ehe Sie sich's versehen, sind Sie zurück auf Feld A1 und in der Anfangsmaske.

Als nächstes verlagern Sie den Cursor auf Feld Y1. Die Eingabe lautet demnach:

[F5] y1 [ENTER]

Im Streifen über dem Raster sehen Sie, daß die Spalten rechts von Z mit Doppelbuchstaben bezeichnet sind, AA, AB, AC. Es geht also nicht etwa mit dem letzten Buchstaben des Alphabets zu Ende. Abacus enthält insgesamt 64 Spalten. Auf AZ folgen die Bezeichnungen BA, BB usw., bis zur letzten Spalte in der Tabelle, BL.

Sie können die Tabelle auch in der anderen Richtung durchschreiten – nur dauert das etwas länger, weil Abacus 255 Zeilen umfaßt.

WERTE EINGEBEN

Setzen Sie den Cursor in Feld A1 und schreiben Sie

100

Warten Sie noch mit dem abschließenden **ENTER**. Im mittleren Feld im Steuerbereich ist jetzt der Hinweis "eintippen und **ENTER** drücken" farblich vom Rest abgehoben, und ferner sehen Sie in der Zeile direkt unter dem Fenster den Vermerk **Wert > 100**.

Jede Eingabe, die Sie vornehmen, und die Meldungen, die Abacus beim Aufrufen von Befehlen ausgibt, werden in dieser Zeile angezeigt. Es ist die *Eingabezeile*. Sie verfügt über eine eigene Schreibmarke, den *Eingabecursor*, der nicht mit dem Abacus-Tabellencursor zu verwechseln ist. Fehler in der Eingabezeile können Sie leicht mit Hilfe des Zeileneditors korrigieren, der in der Einführung zu den QL- Programmen beschrieben ist.

ENTER überträgt den Wert 100 in das aktuelle Feld (A1) und leert die Eingabezeile für neue Eingaben. In der untersten Zeile der Maske erscheint als INHALT von FELD A1 ebenfalls die Zahl 100.

TEXTEINGABE

Text oder Buchstaben werden genauso in Felder eingetragen wie Werte, nur muß jede Texteingabe durch Anführungszeichen (Gänsefüßchen) eingeleitet werden. Sobald Sie das Anführungszeichen drücken, reagiert Abacus durch Hervorhebung des TEXT-Hinweises im Steuerbereich und durch die Anfrage **Text >** " in der Eingabezeile. Geben Sie den gewünschten Text ohne schließendes Anführungszeichen ein und drücken Sie **ENTER**. Füllen Sie noch ein paar Felder mit verschiedenen numerischen und alphabetischen Eingaben und probieren Sie auch den Unterschied aus zwischen:

1000 [ENTER] (als numerische Eingabe)

und

"1000 [ENTER] (Text)

Die Plazierung eines Wertes in einem Feld ist rechts, die eines Textes linksbündig. Außerdem ist in der Rubrik INHALT im Statusbereich ersichtlich, ob es sich um einen Wert oder um Text handelt.

DIE BEFEHLE

Die Abacus-Befehle sind über **F3** erreichbar.

Das Mittelfeld im Steuerbereich präsentiert einen Katalog der verfügbaren Befehle, das *Befehlsmenü*. (Abbildung 2.9)

Die meisten Befehle werden in späteren Kapiteln ausführlich erklärt, doch für den Moment wollen wir uns nur mit zweien befassen: **tilgen** zum Leeren der Tabelle und **verlassen** zum Umsteigen von Abacus auf SuperBASIC.

Probieren Sie als erstes **tilgen**. **F3** und **T** für Tilgen drücken. Sie brauchen nie mehr als den Anfangsbuchstaben eines Befehls einzugeben. Sofort erscheint in der Eingabezeile der angeforderte Befehl, während der Steuerbereich das entsprechende Menü anzeigt. Machen Sie den Befehl mit **ESC** rückgängig.

Wiederholen Sie den Vorgang, aber jetzt drücken Sie nach **F3** und **T ENTER**, worauf das Arbeitsblatt geleert und der Cursor wieder auf seine Ausgangsposition zurückgesetzt wird. Nun kann's von vorn losgehen.

Der Wechsel von Abacus auf SuperBASIC muß immer mit dem **verlassen**-Befehl erfolgen. Auch hier drücken Sie erst **F3** und dann den Anfangsbuchstaben des Befehls, **V**. Die Ausführung des Befehls bewirkt den Verlust aller Ihrer Arbeitsdaten. Im Zweifelsfall kehren Sie mit **ESC** auf die Hauptebene von Abacus zurück.

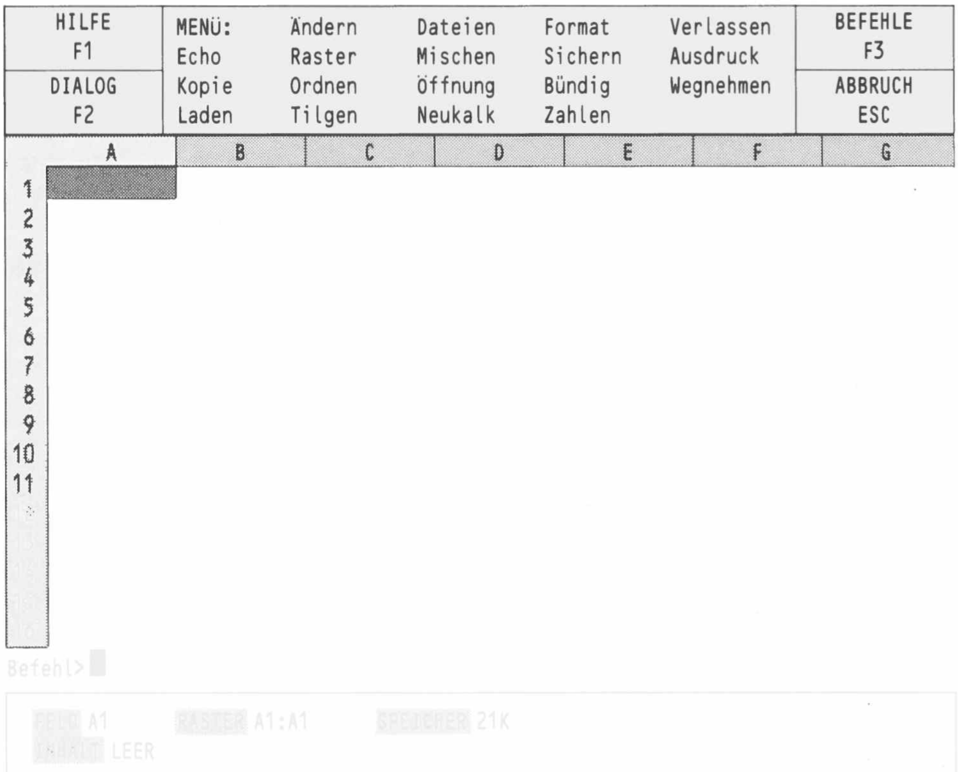


Abbildung 2.9 Das Befehlsmenü

KAPITEL 3

FELDER, ZEILEN, SPALTEN UND BEREICHE

Die Effizienz von Abacus liegt unter anderem in seiner Fähigkeit, ganze Zeilen, Spalten und Zellenbereiche mit einer einzigen Operation zu erfassen. Auf diese Weise ist es z.B. möglich, alle Felder einer Zeile oder eines Bereichs automatisch zu füllen, entweder mit jeweils dem gleichen Inhalt oder mit Werten, die sich nach einer Berechnungsvorschrift systematisch verändern.

Als erstes befassen wir uns mit den Eigenschaften der Felder und der Art und Weise, wie auf Felder verwiesen werden kann.

FELDER

Der Grundbaustein zur Aufnahme und Speicherung von Informationen ist das *Feld* (auch *Zelle* genannt). Jedes Abacus Tabellenfeld kann ein Informationselement aufnehmen, welches entweder ein Stück Text, ein Wert oder eine Formel sein kann.

Zu jedem gefüllten Feld speichert Abacus interne Zusatzinformationen über das Anzeigeformat, etwa ob der Inhalt rechts, linksbündig oder zentriert oder ob ein numerischer Wert im Währungs- oder im Exponentialformat usw. darzustellen ist.

Bündig

Mit dem Befehl **bündig** wird die Position des Inhalts im Feld geändert. Zahlen und Text können links- oder rechtsbündig bzw. mittig angeordnet werden. Der Befehl kann sich auf eine Einzelzelle oder auf einen ganzen Bereich beziehen.

Geben Sie die Zahl 100 im Feld A1 ein, und drücken Sie dann **F3** und **B** für **bündig**. Als erstes muß der Geltungsbereich entschieden werden: er kann sich auf bereits besetzte Felder beschränken, oder er kann zur Standardeinstellung erklärt werden, d.h. für "Leere Felder" gelten. Drücken von **ENTER** wählt die Option "besetzte Felder". Die nächste Auswahl betrifft Text oder Zahlen. Drücken Sie **Z** für Zahlen, worauf im Steuerbereich die drei Optionen **L**inks, **M**ittig und **R**echts angezeigt werden. Wählen Sie mit **ENTER** die Standardvorgabe Links.

Zum Schluß fragt Abacus nach dem Bereich, für den die Änderung Gültigkeit haben soll. Drücken Sie für den Moment einfach **ENTER** für das Einzelfeld A1, dessen Inhalt (100) sich nach links verschiebt.

Erwähnt sei hier noch, daß es möglich ist, das numerische Format und die Bündigkeit eines Feldes zu ändern, welches im Moment gar keinen numerischen Inhalt hat, sondern Text. Allerdings bleibt der Befehl vorläufig wirkungslos und tritt erst in Erscheinung, wenn zu einem späteren Zeitpunkt ein numerischer Inhalt dort eingegeben wird. Analog verhält es sich im umgekehrten Fall, wenn auf ein numerisches Feld eine Textspezifikation angewendet wird.

Leere Felder

Für Abacus existieren Felder ohne Inhalt nicht, und sie beanspruchen daher auch keinen Speicherplatz. Folglich sind sie auch ohne Eigenschaften. Der Versuch, die Befehle **bündig** oder **zahlen** auf ein leeres Feld anzuwenden, bleibt wirkungslos, so daß später dort eingetragene Zahlen im Standardanzeigeformat ausgegeben werden.

Um das Standardformat zu ändern, ist es erforderlich, jeweils die Einstellung der "Leeren Felder" in diesen beiden Befehlen zu modifizieren. Zu diesem Zweck beim **zahlen**-Befehl **F3**, **Z** und **L** drücken und dann z.B. mit **P** und 1 **ENTER** die Darstellung als Prozentsatz mit einer Dezimalstelle zum Standardanzeigeformat erklären. Die verfügbaren Optionen sind gleich den bei der Änderung besetzten Felder, nur daß hier nicht nach dem Bereich gefragt wird.

Die Option "Leere Felder" im Untermenü zum **bündig**- Befehl funktioniert analog. Auch hier ist die Anfrage nach dem Bereich überflüssig, weil das neue Standardformat bei jeder Eingabe in ein leeres Feld zur Anwendung gelangt.

Die neu gesetzten Standardvorgaben bleiben bis zur nächsten Änderung in Kraft bzw. bis zum Verlassen von Abacus und Rückkehr zu SuperBASIC.

Um auf die ursprünglichen Standardwerte umzuschalten – Zahlen: Allgemein und rechtsbündig, Text linksbündig – diese Sequenz eingeben:

[F3] B L Z R	Zahl, rechtsbündig
[F3] B L [ENTER] [ENTER]	Text, linksbündig
[F3] Z L A	Zahl, Allgemein

ZEILEN

Kalkulationstabellen enthalten oft mehrere Felder mit dem gleichen Wert oder mit Werten, die in einer systematischen Weise miteinander verknüpft sind. Abacus verfügt über verschiedene Methoden, um Ihnen solche langwierigen Eingabearbeiten zu ersparen. Zum einen ist es möglich, eine Serie von Zellen mit einem gemeinsamen *Bereichsbezeichner* anzusprechen. Zu diesem Zweck gibt es die beiden Bezeichner **sp** und **zle**, welche die Felder der aktuellen Spalte bzw. der aktuellen Zeile umfassen (d.h. die Zeile oder Spalte, die den Cursor enthält).

Zur Veranschaulichung des Gesagten wollen wir die erste Zeile von Spalte B bis D mit dem Wert 100 füllen. Dazu verwenden wir den Bereichsbezeichner **zle**. Zuerst den Cursor in das Feld A1 setzen und eingeben:

zle = 100 [ENTER]

Sofort nach Drücken von **ENTER** macht Abacus in der Eingabezeile den Vorschlag, mit Spalte A anzufangen (weil dort der Cursor sitzt). **ENTER** würde dies bestätigen. Um mit Spalte B anzufangen,

B [ENTER]

drücken, worauf in der Eingabezeile B als Ausgangsspalte und ein anderes Feld als Ende des Bereichs angezeigt wird, nämlich BL (das letzte Feld in der Tabelle). Geben Sie ein

D [ENTER]

Die auf diese Weise vervollständigte Anweisung wird sofort ausgeführt: die Felder B1, C1 und D1 enthalten jetzt den Wert 100, und die Eingabezeile wird freigegeben.

SPALTEN

Das Ausfüllen einer Spalte folgt genau dem gleichen Muster, nur daß hier der Bereich mit Zahlen, nicht mit Buchstaben markiert werden muß. Nehmen wir an, wir wollen die Zeichenkette "TAG" in alle Felder in Spalte D einsetzen, die im Bereich zwischen Zeile 5 und Zeile 11 liegen. Dazu brauchen wir den zweiten Bereichsbezeichner, **sp**. Setzen Sie den Cursor ins Feld D5 und schreiben Sie:

sp = "TAG" [ENTER]

Abacus schlägt Zeile 5 als Anfangspunkt vor, da sich der Cursor dort befindet. Bestätigen Sie dies mit **ENTER**. Der nächste Vorschlag ist Zeile 255, was Sie mit

11 [ENTER]

richtigstellen. Nun erscheint der eingegebene Text in allen Feldern zwischen D5 und D11, und die Eingabezeile wird freigegeben.

Bei jedem Gebrauch von **sp** offeriert Abacus einen Ausgangs– und einen Endpunkt für die Operation, den Sie entweder durch bloße Eingabe von **ENTER** akzeptieren oder durch eine eigene Spezifikation überschreiben.

Außer der eben beschriebenen Anwendung von **sp** und **zle** können diese beiden Bezeichner allen Funktionen beigegeben werden, die einen Bereich als Parameter verlangen. Dazu ein Beispiel zum Ausprobieren:

Füllen Sie die obere linke Ecke des Fensters von A1 bis C3 (also 3 mal 3 Felder) mit beliebigen Zahlen. Verlagern Sie dann den Cursor ins Feld D1, wo Sie eingeben:

sp = summe(zle) [ENTER]

Anders ausgedrückt: jedes Feld der Spalte D ist gleich der Summe der Werte in den Feldern der entsprechenden Zeile. Abacus braucht dazu die Spalten– und Zeilenbereiche. Das zweimal vorgeschlagene D muß ersetzt werden durch A als Ausgangspunkt und C als Endpunkt des Zeilenbereichs. Nach jeder Eingabe **ENTER** drücken. Bei der Abgrenzung des Spaltenbereichs können die vorgeschlagenen Ziffern 1 und 3 akzeptiert werden. Abacus addiert die Werte in horizontaler Richtung und gibt in den Feldern der Spalte D die Resultate aus.

NAMEN

In den vorangehenden Beispielen wurden ausdrückliche Feldadressen (Spalten– und Zeilen-Koordinaten) verwendet. Eine wichtige alternative Methode zur Bezeichnung von Feldern verwendet stattdessen *Namen*, die Sie sich frei ausdenken können. Diese Namen dienen dann ganz genauso zur eindeutigen Identifizierung von Zeilen und Spalten oder einzelnen Zellen.

Jeder Text, d.h. jede Zeichenkette in einem Feld kann als dessen Name fungieren. Und dieser Name kann überall dort eingesetzt werden, wo sonst Koordinaten-Feldadressen verwendet werden, also z .B. in Befehlen und Formeln. Der Vorteil dabei ist, daß es viel leichter ist, sich sinnvolle Namen zu merken als irgendwelche Zahlen und Buchstaben.

Außerdem ist es eine äußerst effiziente und flexible Methode, die das Einrichten und die Funktionstüchtigkeit Ihrer Tabelle in vieler Hinsicht erleichtert. Die nächsten beiden Abschnitte erklären den Einsatz von Namen.

Namen für Zeilen und Spalten

Ob sich ein Name auf eine Zeile oder auf eine Spalte bezieht, hängt vom Inhalt der übrigen Felder in der Tabelle ab. Die Faustregel bei der Namengebung ist, daß Abacus ausgehend von dem Namenfeld nach unten und nach rechts sucht.

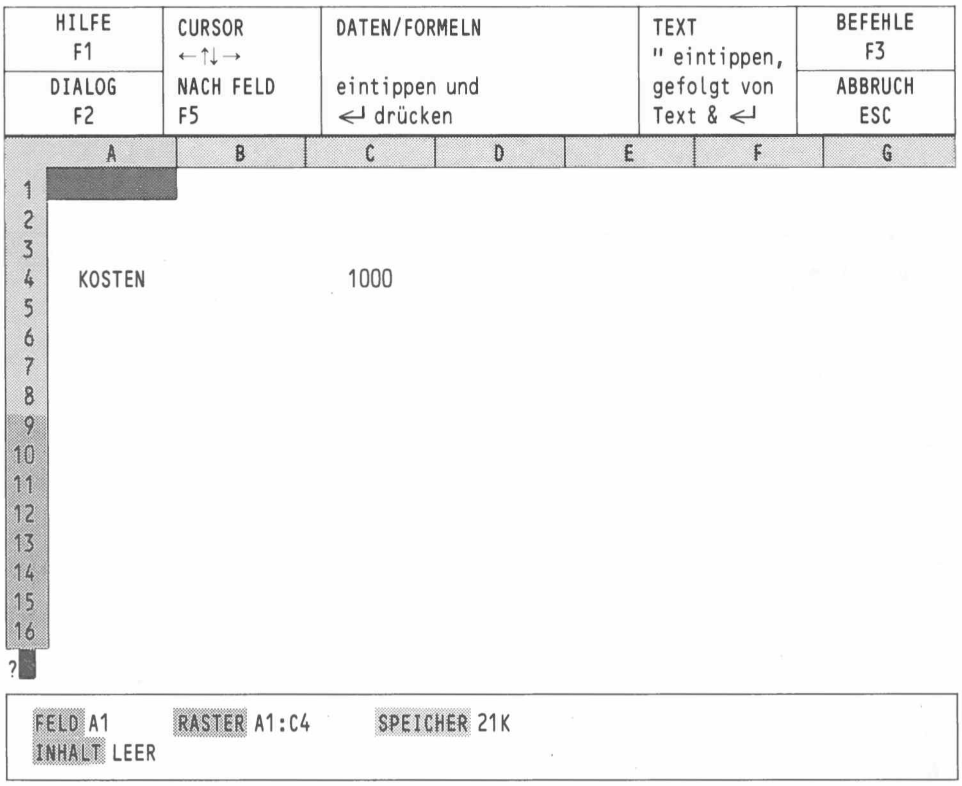


Abbildung 3.1 Benennen einer Zeile

Das jeweils nächstgelegene numerische Feld nach unten oder nach rechts bestimmt, ob sich der Name auf eine Zeile oder auf eine Spalte bezieht. Diese etwas abstrakte Erläuterung ist anhand der beiden Abbildungen 3.1 und 3.2 leicht nachzuvollziehen. In Abbildung 3.1 bezeichnet der Name eine Zeile, in Abbildung 3.2 eine Spalte.

In komplizierteren Fällen, wenn rechts und unterhalb des Namens Werte stehen, ist das Feld, das durch die geringste Anzahl von Feldern von dem Namenfeld getrennt ist, ausschlaggebend. Wenn der Abstand identisch ist, und der Name in einer Formel eingesetzt wird, beschwert sich Abacus mit der lakonischen Meldung:

```
Name ist weder Zeile noch Spalte
```

und wartet, bis Sie die Leertaste drücken und anschließend den zweideutigen Namen in der Eingabezeile mit **sp** bzw. **zle** spezifizieren. Es empfiehlt sich, Namen so anzuordnen, daß Abacus sie als zu einer Zeile oder Spalte gehörend identifizieren kann.

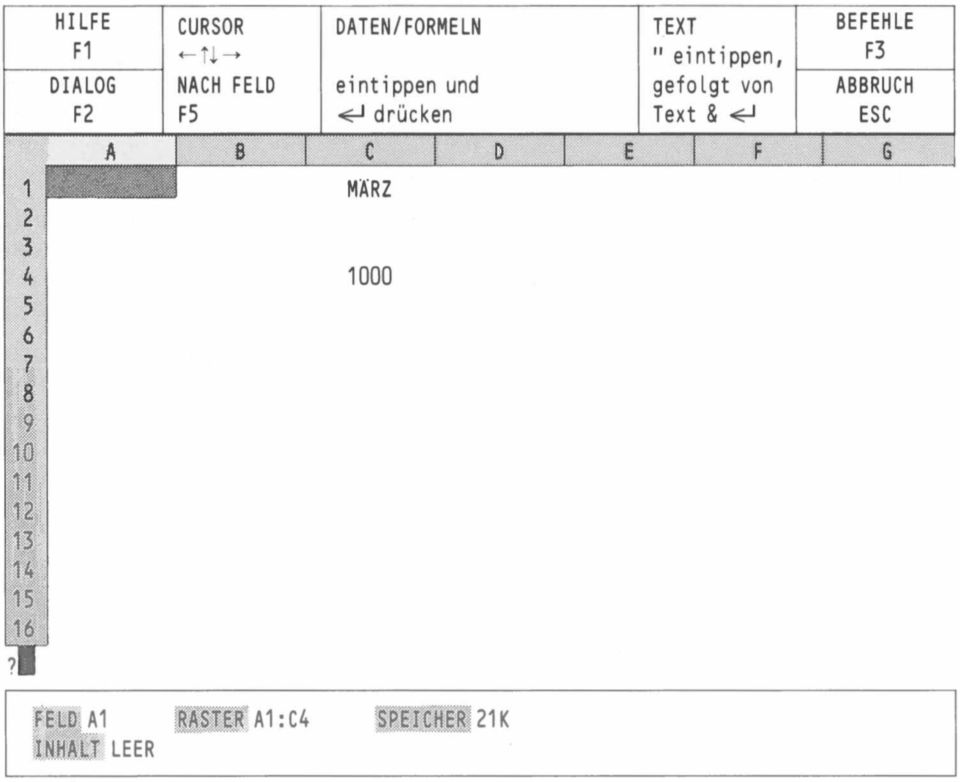


Abbildung 3.2 Benennen einer Spalte

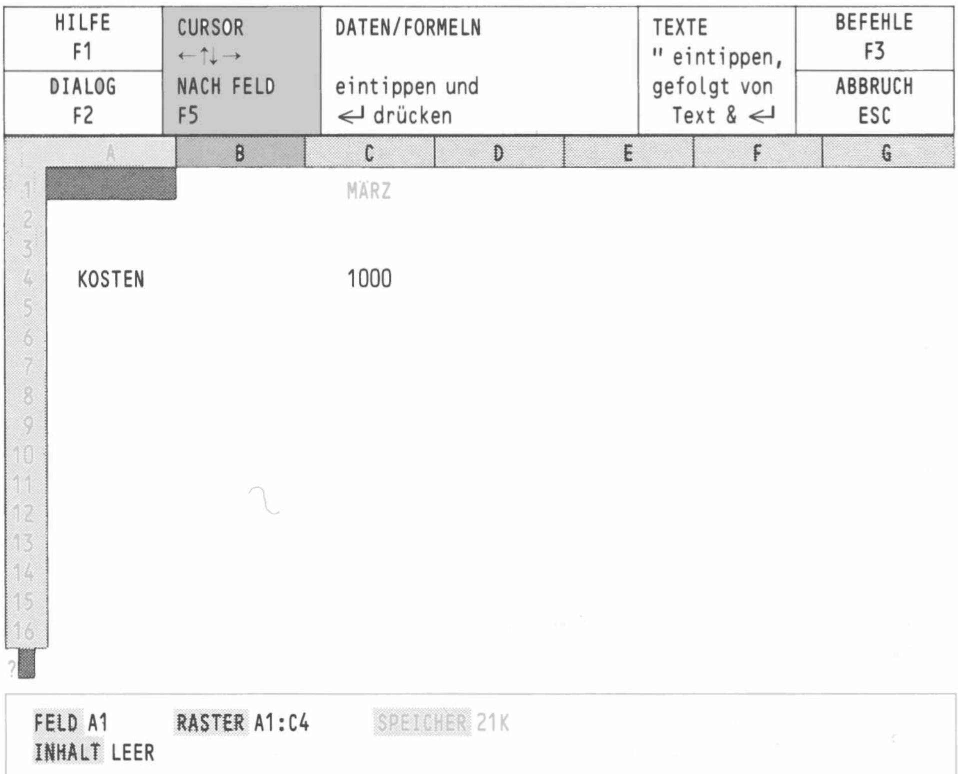


Abbildung 3.3 Benennen eines Feldes

Felder benennen

Auch Einzelfelder können mit Namen angesprochen werden, statt in Form von Koordinaten. Allerdings braucht man dazu immer zwei Namen. So etwa in der Abbildung "März" und "Kosten" als Adresse für Feld C4.

Die Feldadresse besteht aus den Namen der Spalte und der Zeile, in deren Schnittpunkt das Feld liegt, mit einem Punkt als Begrenzer dazwischen, z.B. März.Kosten. Die Namen brauchen nicht ausgeschrieben zu werden, solange sie eindeutig sind, und sie können groß oder klein geschrieben werden. In unserem Beispiel wäre z .B. die Angabe "Mär.Kos" mehr als ausreichend. Auch die Reihenfolge ist unwichtig, so daß die Adresse auch mit "Kos.Mär" angegeben werden könnte.

BEREICHE

Befehle können sich nicht nur auf Zeilen und Spalten beziehen, sondern auch auf rechteckige Tabellenblöcke, sog. *Bereiche*.

Ein Verweis auf einen Bereich setzt sich aus zwei Elementen zusammen: den Koordinaten des oberen linken Feldes und den Koordinaten des unteren rechten Feldes in dem betreffenden Block. Die beiden Teile sind durch Doppelpunkt voneinander zu trennen. Ein Beispiel eines Bereichsverweises ist:

```
A2:D27
```

Ein typisches Anwendungsbeispiel für einen solchen Zellenbereich ist der **kopie**-Befehl, mit dem der Inhalt eines ganzen Feldblocks an eine andere Position in der Tabelle kopiert wird.

Viele Befehle fordern einen Bereichsverweis an, um ihr Geltungsgebiet zu identifizieren. Da Ihnen bei der Definition eines Bereichs sehr viel mehr Flexibiität gegeben ist als bei Spalten oder Zeilen, sieht Abacus hier davon ab, einen Vorschlag zu machen, und wartet, bis Sie den ganzen Verweis eingeben. Dieser kann vier verschiedene Formen annehmen:

1. Mit ausdrücklichen Zeilen– und Spaltenkoordinaten (Zahlen und Buchstaben): z.B. A1:C7
2. Mit Namen:
   z.B. Januar.Verkauf:März.Kosten
3. Mit einer Kombination dieser beiden Methoden:
   z.B. A1:März.Kosten
4. Mit einem Bereichsbezeichner:
   z.B. zle (oder sp)
   Diese Angabe bezieht sich auf die Felder der Zeile bzw. Spalte, die den Cursor enthält. In diesem Fall macht Abacus einen Vorschlag für einen geeigneten Anfangs– und Endpunkt.

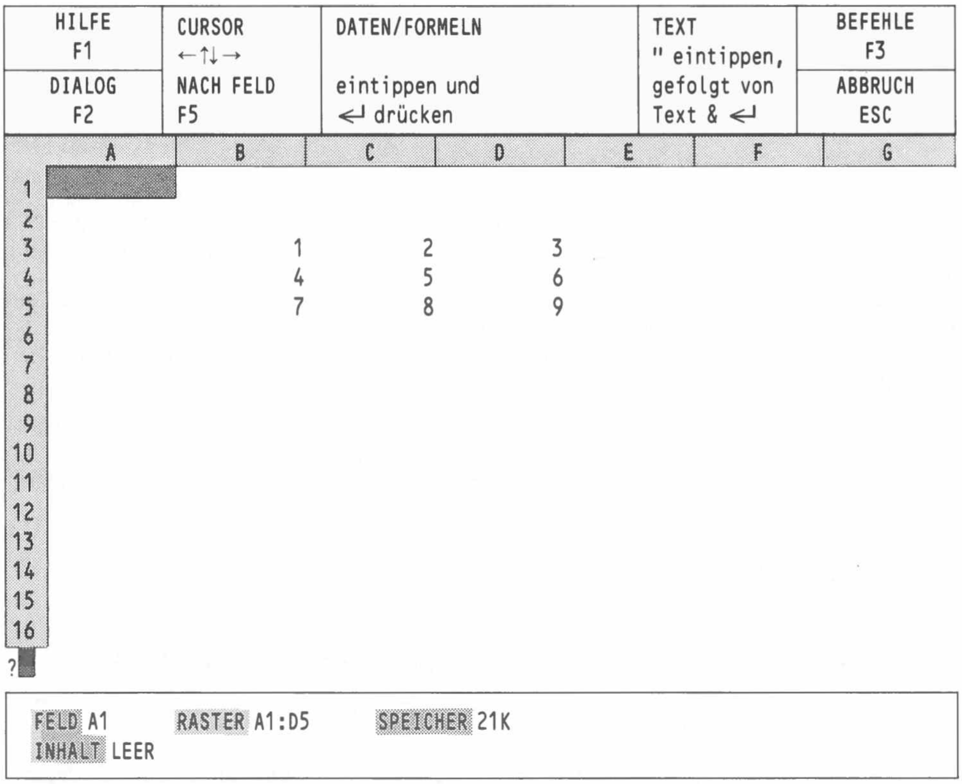


Abbildung 3.4 Verweis auf einen Feldbereich

MEHR ZU WERTEN UND TEXTEN

Die Angabe der Position, d.h. der Adresse eines Einzelfeldes oder auch eines ganzen Feldbereichs dürfte keine Schwierigkeiten mehr machen. Nun wollen wir sehen, wie die Darstellung der Feldinhalte verändert werden kann. Zu diesem Zweck als erstes eine kurze Erklärung über die Art und Weise, wie Zahlen gespeichert werden. Verlagern Sie den Cursor auf A1, wo Sie den Wert 123.456 eingeben. An dieser Stelle sei darauf hingewiesen, daß in Abacus immer ein *Dezimalpunkt* zu verwenden ist, kein Komma. Wo von "Nachkommastellen" die Rede ist, bedeutet das im Klartext "Ziffern nach dem Punkt".

Abacus speichert alle Zahlen mit einer Genauigkeit von intern 16, extern 14 Ziffern, d.h. 14 Ziffern werden angezeigt, wobei die beiden zusätzlichen sicherstellen, daß die Zahl exakt ausgegeben wird. Obwohl Abacus alle Speicher– und Rechenvorgänge mit dieser eindrucksvollen Genauigkeit ausführt, brauchen Sie sich nicht alle Ziffern anzeigen zu lassen.

Wählen Sie den **zahlen**-Befehl mit **F3** und **Z**. Zwei Optionen stehen zur Wahl: Besetzte oder Leere Felder. Drücken von **ENTER** bringt Sie zu "Änderung Besetzter Felder".

Abacus stellt Ihnen verschiedene Formate zur Wahl:

Drücken Sie **W** für Währung (Anzeige als Geldbetrag). Abacus fragt nach der gewünschten Darstellung von negativen Werten und schlägt das Minuszeichen vor, was Sie mit **ENTER** akzeptieren können. Die Alternative ist die in der Buchhaltung übliche Einklammerung negativer Werte, die Sie bei Bedarf mit **K** anwählen. Hier spielt es keine Rolle, für welche Anzeigeform Sie sich entscheiden; wir nehmen die Standardvorgabe (Minuszeichen) an.

Nun fordert Abacus die Eingabe des Bereichs, für den das neue Anzeigeformat Gültigkeit haben soll. Schreiben sie einen entsprechenden Verweis, z .B. A1:B3 oder auch nur eine einzige Feldadresse, worauf Abacus einen Endpunkt vorschlägt. In manchen Fällen wird das A1:A1 sein, was natürlich identisch mit der Einzelfeldadresse A1 ist. Akzeptieren Sie mit **ENTER** bzw. überschreiben Sie die Vorgabe mit dem gewünschten Verweis, gefolgt von **ENTER**.

Die gesamte Eingabesequenz sieht so aus (wenn Sie den Bereichsverweis A1:A1 akzeptieren):

[F3] z [ENTER] w [ENTER] [ENTER]

Vor dem letzten **ENTER** zur Ausführung des Befehls lautet der Inhalt der Eingabezeile wie folgt:

```
Befehl>Zahlen,besetzt,Währung,Minuszeichen,Bereich A1:A1
```

Nach dem letzten **ENTER** ändert sich die Anzeige von Feld A1 auf $123.46, während der tatsächliche Inhalt des Feldes unverändert 123.456 bleibt, wie Sie unten im Statusbereich erkennen können. Abacus bringt Sie automatisch zurück zur Hauptanzeige.

Das "Währungsformat" rundet stets auf zwei Dezimalstellen und setzt dem Betrag ein Währungssymbol voran. Dieses Symbol kann mit Hilfe des **format**-Befehls beliebig geändert werden, darf jedoch immer nur aus einem Symbol bestehen, also einfach "M" für DM usw.

Als nächstes wollen wir dem Feld A1 ein ganzzahliges Format zuweisen. **F3** und **Z** drücken, dann **G** wählen. Auch hier gibt es wieder zwei Optionen für die Darstellung negativer Werte. Zur Abwechslung schreiben wir **K** für Klammern, gefolgt von **ENTER**.

Die vollständige Eingabesequenz lautet demnach:

[F3] z [ENTER] G K [ENTER]

und der Inhalt der Eingabezeile:

```
Befehl>Zahlen,besetzt,Ganzzahl,Klammern,Bereich A1:A1
```

Den Befehl mit **ENTER** abschließen, worauf im Feld A1 der ganzzahlige Wert (ohne Dezimalstellen) 123 erscheint.

Nun zum Dezimalformat. Hier und bei den restlichen Formaten wird auf die Optionen zur Darstellung negativer Werte verzichtet, dafür muß die Zahl der Nachkommastellen angegeben werden. Nehmen wir einmal fünf Dezimalstellen an. Rufen Sie den Befehl wieder mit **F3** und **Z** auf, und bestätigen Sie das Standardformat **D** (Dezimal) mit **ENTER**. Überschreiben Sie den Standardwert 2 mit 5 und akzeptieren Sie den vorgeschlagenen Bereich A1:A1 mit **ENTER**. Die vollständige Eingabesequenz lautet also:

[F3] z [ENTER] [ENTER] 5 [ENTER] [ENTER]

und der Inhalt der Eingabezeile:

```
Befehl>Zahlen,besetzt,Dezimal,Dezimalstellen 5,
     Bereich A1:A1
```

Feld A1 zeigt den Wert 123.45600.

Als nächstes das Prozentformat, das Sie im Untermenü mit **P** erreichen. Überschreiben Sie die vorgeschlagene 2 für Dezimalstellen mit 1 und akzeptieren Sie A1:A1.

Die vollständige Eingabesequenz und die zugeörige Eingabezeile sind:

[F3] Z [ENTER] P 1 [ENTER] [ENTER]

```
Befehl>Zahlen,besetzt,Prozentsatz,Dezimalstellen 1,
     Bereich A1:A1
```

Die Anzeige im Feld A1 ist jetzt 12345.6%; das Prozentformat multipliziert die Zahl mit 100 und fügt ein Prozentzeichen hinzu. Unabhängig davon ist der tatsächliche Inhalt des Feldes nach wie vor 123.456 (siehe Statusbereich).

Versuchen wir es als nächstes mit dem Exponentialformat, wobei wir drei Dezimalstellen vorsehen. Eingabesequenz:

[F3] Z [ENTER] E 3 [ENTER] [ENTER]

Die entsprechende Eingabezeile lautet:

```
Befehl>Zahlen,besetzt,Exponent,Dezimalstellen 3,
     Bereich A1:A1
```

Sobald Sie den Befehl mit einem dritten **ENTER** abschließen, erscheint die Zahl im Feld A1 in der Exponentialschreibweise: 1.235E+02 (1.235 ist die Mantisse, die auf den Buchstaben E folgende Zahl ist der Exponent, der die Zehnerpotenz angibt, mit der die Mantisse zu multiplizieren ist).

Die Exponential- oder wissenschaftliche Notation ist eine Kurzschreibweise für sehr große oder sehr kleine Zahlen, die sich nicht im Dezimalformat darstellen lassen. Die Zahl wird als ein Wert zwischen 1 und 10 geschrieben und mit der entsprechenden Zehnerpotenz multipliziert. Die Zahl 2 300 000 000 z.B. kann geschrieben werden als 2.3 multipliziert mit 1 000 000 000, wobei 1 000 000 000 eins hoch 9 ist. 2 300 000 000 läßt sich folglich im Exponentialformat als 2.3E+09 ausdrücken. Sehr kleine Werte werden mit negativen Zehnerpotenzen dargestellt, so daß die Zahl 0.000123 – d.h. 1.23 dividiert durch 10 000 (zehn hoch vier) – als 1.23 E–04 geschrieben werden kann.

Bleibt uns noch das Allgemeine Zahlenformat, welches Sie wie folgt anfordern:

[F3] Z [ENTER] A [ENTER]

wobei die Eingabezeile

```
Befehl>Zahlen,besetzt,Allgemein,Bereich A1:A1
```

enthält. Auch hier wird auf die Anfrage nach den Stellen nach dem Komma (bzw. nach dem Punkt) verzichtet. Abacus versucht bei Anzeige im Allgemeinen Format jeweils die optimale Lösung zu finden, d.h. den Wert im Rahmen des verfügbaren Platzes möglichst genau wiederzugeben.

Bevor wir uns vom Befehl **zahlen** etwas anderem zuwenden, noch ein kleines Experiment. Versuchen Sie mit folgender Eingabe ein Dezimalformat mit 9 Stellen anzufordern:

[F3] Z [ENTER] [ENTER] 9 [ENTER] [ENTER]

In Feld A1 erscheint die Anzeige: ##########, als Warnung, daß das gewünschte Anzeigeformat nicht in die vorgesehenen Felder paßt. In solchen Fällen sollte entweder das Anzeigeformat oder die Breite der betreffenden Spalte modifiziert werden.

Leeren Sie Ihre Tabelle jetzt mit **tilgen**. Geben Sie im Feld A1 diesen Text ein:

```
"Das ist ein langes Stück Text
```

Der Text wird angezeigt und erstreckt sich über mehrere Tabellenfelder. Setzen Sie jetzt den Cursor auf B1 und tragen Sie dort den Wert 1 ein. Jetzt wird der überlange Tex am Ende von A1 abgezwackt; wenn Sie jedoch mit dem Cursor zurück auf A1 gehen, erkennen Sie in der Rubrik "INHALT" im Statusbereich, daß tatsächlich noch immer der ganze Text gespeichert ist.

Verlagern Sie den Cursor wieder auf B1 und fordern Sie aus dem Befehlsmenü (**F3**) **W** für **wegnehmen** an, worauf Sie nach dem Bereich gefragt werden. Akzeptieren Sie das angebotene B1 mit **ENTER**. Die ganze Eingabesequenz lautet:

[F3] W [ENTER]

Der Text in A1 kann sich jetzt wieder breitmachen, da Feld B1 leer ist.

KAPITEL 4
FUNKTIONEN UND FORMELN

FUNKTIONEN

Abacus verfügt über eine reichliche Auswahl von *Funktionen* zur Ausführung gewisser Berechnungen an Feldern oder Feldbereichen. Funktionen arbeiten mit mehreren Eingabewerten, sog. *Argumenten*, anhand derer sie ein Ergebnis ermitteln. Man sagt, das Ergebnis sei der *Wert, den die Funktion liefert* oder einfach der *Funktionswert*.

Abacus verlangt die Eingabe der Argumente in Klammern nach dem Namen der Funktion. Mehrere Argumente werden durch Kommas voneinander abgegrenzt. Die meisten Abacus-Funktionen liefern als Ergebnis einen numerischen Wert, etwa die Funktion **summe()**, welche als Argument einen Bereichsverweis akzeptiert und als Resultat die Summe aller Feldinhalte in dem betreffenden Bereich ausgibt.

Einige andere Funktionen, z.B. **monat()**, liefern als Resultat einen Textwert. **monat(1)** beispielsweise ergibt "Januar". Einige wenige Funktionen kommen ohne Argumente aus – dennoch muß ein leeres Klammerpaar angegeben werden. Zum Beispiel gibt die Funktion **pi()** den Wert der mathematischen Konstanten π ($\approx$ 3.14) aus.

Zwei ganz besonders praktische Funktionen sind **sp()** und **zle()**. Sie liefern die Nummer der Spalte (bzw. Zeile), an deren Schnittpunkt das Feld mit der Funktion liegt. Die Beispiele im nächsten Kapitel machen ausgiebig von diesen zwei Funktionen Gebrauch.

sp() gibt die Zahl 1 für Spalte A aus, 2 für Spalte B usw., während **zle()** ganz einfach die effektive Zeilennummer liefert.

Hier ein praktisches Beispiel. Wir benutzen die beiden Funktionen **monat()** und **sp()**, um die Tabellenspalten mit Überschriften zu versehen, d.h. die Spalten B bis M sollen mit den Monatsnamen versehen werden. Die Funktion **sp()** liefert uns den Wert, den **monat()** als Argument benötigt, um in jeder Spalte ein anderes Ergebnis zu errechnen. Geben Sie ein:

```
zle = monat(sp())
```

und drücken Sie **ENTER**. Wählen Sie als Bereich Spalte B bis M. Das Ergebnis entspricht nicht ganz unseren Vorstellungen: der erste Monat in Spalte B ist Februar, statt Januar. Der Grund liegt darin, daß die Funktion **sp()** für Spalte B den Wert 2 (nicht 1) ausgibt. **monat()** mit dem Argument 2 liefert dann eben Februar. Folglich müssen wir unsere Anweisung ein wenig abändern, nämlich so, daß vom Wert von **sp()** 1 subtrahiert wird, bevor er als Argument für **monat()** eingesetzt wird:

```
zle = monat(sp()-1)
```

Vergessen Sie nicht, jede Eingabe mit **ENTER** abzuschließen. Wählen Sie auch hier wieder den Spaltenbereich B bis M.

FORMELN

In Abacus dienen Formeln im allgemeinen dazu, den Inhalt eines Feldes mit einem oder mehreren anderen Feldern zu verknüpfen. Die Fähigkeit, solche Formeln zu verarbeiten, trägt ganz erheblich zur Flexibilität und zur Effizienz von Abacus bei. Selbst die kompliziertesten Zusammenhänge lassen sich auf diese Weise einfach ausdrücken.

Formeln werden genau wie Werte eingegeben: durch Verlagern des Cursors in das betreffende Feld, Schreiben der Formel und abschließendes Drücken von **ENTER**. Für Abacus ist jede Eingabe, die nicht als Wert (erstes Zeichen eine Zahl) oder als Text (erstes Zeichen Anführungszeichen) interpretiert werden kann, eine Formel.

Tragen Sie im Feld B3 100 und im Feld C3 200 ein; verlagern Sie dann den Cursor auf D3 und geben Sie diese Formel ein:

```
B3 + C3
```

Nach Drücken von **ENTER** beobachten Sie zwei Dinge: einmal erscheint im Feld D3 die Zahl 300 als Summe der beiden Felder B3 und C3, zum andern wird im Statusbereich als Inhalt des aktuellen Feldes D3 nicht das Ergebnis der Formel angezeigt, sondern die tatsächlich zugrundeliegende Formel.

Ein Tabellenfeld, welches eine Formel enthält, zeigt immer das Ergebnis der Kalkulation, während die zugrundeliegende Formel der Rubrik "INHALT" im Statusbereich zu entnehmen ist.

Die restlichen Beispiele, die mit Formeln arbeiten, verwenden die Namen-Funktion und die Zeilen- und Spaltenbereichsbezeichner. Sie ermöglichen sehr viel effizientere Methoden zur Informationseingabe als die Koordinatenverweise.

Zur Beachtung: gewöhnliche numerische Formeln, die keine Verweise auf Felder enthalten, speichert Abacus nicht als Formeln, sondern einfach als Resultat der Berechnung. So wird z .B. 37 + 100/20 als 42 gespeichert und nicht als die eingegebene Formel.

Einfache Cash-Flow-Analyse

A	B	C	D	E
	Januar	Februar	März	April
Verkauf	1000.00	1050.00	1102.50	1157.63
Kosten	722.00	749.50	778.38	808.69
Gewinn	278.00	300.50	324.13	348.93

Abbildung 4.1 Einfache Cash-Flow-Analyse

Leeren Sie Ihr Arbeitsblatt mit **tilgen**, und beginnen Sie den Aufbau der Tabelle mit den Monatsnamen in den Spalten B1 bis M1.

Geben Sie dann in Feld A2 den Text "Verkauf" ein und den Wert 1000 in Feld B2. Mit dem Cursor in Feld C2 schreiben Sie die folgende Formel:

```
zle = Verkauf.Januar*1.05
```

Akzeptieren Sie den von Abacus vorgeschlagenen Bereich (Spalte C bis M) durch zweimaliges Drücken von **ENTER**. (Abacus wählt richtig, weil es davon ausgeht, daß die beiden Bereiche eine identische Länge haben.) Beim zweiten **ENTER** erscheint eine ganze Serie von Werten in der zweiten Zeile, angefangen von Spalte C, und im Statusbereich erkennen Sie die Formel B2 * 1.05.

Wenn Sie mit dem Cursor der Zeile 2 entlang fahren, bemerken Sie, daß die Formel sich verändert. Es wird immer der direkt vorangehende Wert mit 1.05 multipliziert und das Ergebnis in das Tabellenfeld gesetzt. Beispielsweise enthält die Formel in Feld E2 einen Verweis auf D2 und die Formel in H2 einen auf G2 usw.

Alle Formeln in Abacus funktionieren in dieser Weise, es sei denn, Sie verlangen ausdrücklich etwas anderes. Jede Formel berücksichtigt die relativen Positionen der Bezugsfelder im Verhältnis zum eigenen Standort. Beim Einsetzen der gleichen Formel in andere Felder werden die Verweise so modifiziert, daß die *relativen Positionen* bestehen bleiben.

Interessant zu wissen ist vielleicht noch, daß die Eingabe des Ausgangswertes von 1000 aus zwei Gründen erforderlich war, erstens zur eindeutigen Identifikation von "Verkauf" als ein Zeilenname und zweitens zur Spezifikation des ersten durch die Formel zu verwendenden Wertes.

Verlagern Sie jetzt den Cursor auf A3, wo Sie den Text "Kosten" eingeben. Ohne den Cursor weiterzubewegen schreiben Sie die Formel:

```
Kosten = Verkauf * 0.55 + 172
```

Diese Formel kalkuliert die Kosten aus zwei Komponenten, den Herstellungskosten (55% des Verkaufspreises) und den festen Kosten von insgesamt M 172.00.

Akzeptieren Sie den vorgeschlagenen Bereich, eingegrenzt durch die Spalten B und M. Da die Kosten-Zeile mit einem Verweis auf "Verkauf" definiert wird, nimmt Abacus an, daß beide Male derselbe Bereich gelten soll.

Verfolgen Sie auch hier wieder mit Hilfe des Cursors die Veränderung der Formel in den einzelnen Feldern. Auf diese Weise ist sofort einsichtig, wie die Ergebnisse zustandekommen.

Tragen Sie zum Schluß den Text "Gewinn" ins Feld A4 ein, zusammen mit einer weiteren Formel:

```
Gewinn = Verkauf-Kosten
```

Akzeptieren Sie auch hier wieder denselben Bereich (B bis M) und überlassen Sie alles weitere den Rechenkünsten von Abacus. Damit sind wir am Ende dieses einfachen, aber vollständigen Beispiels. Wenn Sie jetzt noch das Anzeigeformat "Währung" über **F3** und **Z**

```
Befehl>Zahlen,besetzt,Währung,Minuszeichen,Bereich B2:M4
```

setzen, dann sollte Ihr Tabellenfenster der Abbildung 4.1 entsprechen.

AUTO-KALKULATION

Nehmen Sie jetzt an dem vorhin eingegebenen Beispiel einer Cash-Flow-Analyse eine Änderung am Verkaufswert in Feld B2 vor (Verkauf.Januar).

Eine schnelle Methode, den Cursor dorthin zu verlagern, ist durch Drücken von **F5** und Eingabe der Feldadresse (entweder B2 oder ver.jan) gefolgt von **ENTER**. Schreiben Sie eine beliebig hohe Verkaufsziffer, und beobachten Sie, welche Wirkung das Drücken von **ENTER** auf die übrigen Werte hat.

Jede Eingabe oder Veränderung, die Sie an einem Feld vornehmen, bewirkt eine automatische Neukalkulation aller Formeln im ganzen Arbeitsblatt. In unserem Beispiel hängen alle Formeln direkt oder indirekt mit dem Wert in B2 zusammen – folglich ändern sich auch alle Feldinhalte in Übereinstimmung mit der Änderung in B2. Zur Erinnerung sei darauf hingewiesen, daß wir davon ausgegangen sind, daß die Verkaufszahlen sich monatlich um 5% erhöhen, wobei der Basiswert der Januar–Verkauf ist.

Diese automatische Kalkulation kann mit dem **format**–Befehl auch ausgeschaltet werden, was sich besonders dann empfiehlt, wenn das Arbeitsblatt viele komplexe Formeln enthält, welche die Verarbeitungsgeschwindigkeit beeinträchtigen. Es hat keinen Sinn, bei jeder Feldänderung eine Neukalkulation abzuwarten.

Den **format**-Befehl erreichen Sie über **F3** und **F**. Sofort erscheint in der Maske das Format-Menü, welches Sie in Abbildung 4.2 sehen. Jede Auswahlzeile kann durch Eingabe des Anfangsbuchstabens aktiviert, d.h. zur Veränderung freigegeben werden. Auto-Kalkulation wählen Sie durch Eingabe von **A**, worauf sofort von "ja" auf "nein" umgeschaltet wird. Zum Verlassen des Befehls **ENTER** drücken.

HILFE F1	FORMAT erlaubt Änderung der Optionen Den 1. Buchstaben der Optionen drücken	BEFEHLE F3
DIALOG F2	Für Ende ↵ drücken	ABBRUCH ESC

```
AUTO-KALKULATION bei Eingabe.......................................... ja
LEER wenn Null........................................................ nein
BERECHNUNGSREIHENFOLGE Zle/Spalte .................................... ZLE
80,64,40 Spalten-ANZEIGE (8, 6, 4) ................................... 80
FORMULAR-Vorschub zwischen Seiten .................................... ja
ZEILENABSTAND beim Ausdruck........................................... 0
SEITENLÄNGE Druckerpapier (Zeilen).................................... 64
WÄHRUNGS-Symbol (z.B. f,$) ........................................... $
DRUCKERPAPIER-Breite (Zeichen)........................................ 80
```

Abbildung 4.2 Das Menü des Format-Befehls

Eine in diesem Zustand vorgenommene Veränderung des Inhalts von B2 hat keine Auswirkungen auf andere Felder.

Hingegen können Sie jederzeit eine Neukalkulation aller Formeln im Arbeitsblatt anfordern, und zwar mit **neukalk**. Voraussetzung ist, daß der Auto-Kalkulationsmodus ausgeschaltet wurde. Rufen Sie das Befehlsmenü mit **F3** und mit **N** die Neukalkulation. Sofort erfolgt eine Neuberechnung aller Werte im Arbeitsblatt.

Bevor Sie sich an die nun folgenden Übungsbeispiele machen, sollten Sie den Auto-Kalkulationsmodus wieder einschalten. Fordern Sie aus dem Befehlsmenü mit **F** den **format**-Befehl und dort mit **A** "Auto-Kalkulation" an. Verlassen des Befehls mit **ENTER**.

KAPITEL 5
BEISPIELE

In diesem Kapitel wollen wir einige Übungstabellen aufbauen, um den Gebrauch und die Möglichkeiten von Abacus nicht nur theoretisch zu erläutern, sondern auch in Form von praktischen Beispielen zu veranschaulichen. Bei der Auswahl der Beispiele wurde zum einen darauf geachtet, eine möglichst große Vielfalt der Befehle und Arbeitsvorgänge aufzunehmen und zum anderen einen Einblick in die vielfältigen Anwendungsmöglichkeiten und Einsatzgebiete von Abacus zu vermitteln. Am besten lernt man Abacus kennen, wenn man damit umgeht. Genau dabei wollen Ihnen die folgenden Beispiele helfen.

Wir empfehlen Ihnen, die Beispiele nicht nur als Trockenübung auf dem Papier zu studieren, sondern sie auch selbst am Bildschirm einzugeben und mitzuverfolgen. Neben den eigentlichen Arbeitsanleitungen enthalten sie zusätzliche Informationen und Tips zu bereits früher besprochenen Themen. Vielleicht denken Sie sich auch schon Verbesserungsvorschläge und Anwendungsmöglichkeiten für Ihre eigenen Ansprüche aus.

In den Beispielen werden Text, Werte und Formeln ganz genau so dargestellt, wie sie über die Tastatur einzugeben sind. Einzugebende Feldbereiche werden am Schluß der Zeile in geschweiften Klammern vermerkt. In vielen Fällen stimmt der gewünschte Bereich mit dem von Abacus vorgesehenen überein, so daß Sie keine Eingabe vorzunehmen brauchen, sondern lediglich **ENTER** drücken; andernfalls geben Sie den Bereich von Hand ein. Wenn es erforderlich ist, den Cursor in ein bestimmtes Feld zu plazieren, wird die entsprechende Feldadresse zu Anfang der Zeile in eckigen Klammern angezeigt. Diese sind also nicht einzugeben. Zum Beispiel ist die Angabe

[A4] `zle=monat(sp()-1)` {Spalten B bis M}

wie folgt zu interpretieren:

> Cursor auf Feld A4 verlagern, dann dort
> `zle=monat(sp()-1)` eingeben,
> gegebenenfalls den von Abacus vorgeschlagenen Bereich mit B und M überschreiben.

Wo ein ausdrücklicher Bereichsverweis gefordert wird, z.B. b3:e15, wird exakt diese Form gewählt.

Wenn bei Befehlen die ausführliche Form gewählt wird, entspricht dies der Ausgabe am Bildschirm. Sie brauchen jedoch lediglich den Anfangsbuchstaben einzugeben, bzw. den Vorschlag mit **ENTER** zu akzeptieren.

Bevor Sie ein neues Beispiel in Angriff nehmen, sollten Sie das Arbeitsblatt jeweils mit **tilgen** leeren.

CASH-FLOW BERECHNUNG

Hier handelt es sich um eine etwas komplexere Variante des Beispiels in Kapitel 4. Die fertig erstellte Tabelle sollte Abbildung 5.1 entsprechen.

Als erstes wollen wir dem Arbeitsblatt eine doppelt unterstrichene Überschrift geben:

[C1] `"CASH-FLOW`
[C2] `wiederh("=",länge(c1))`

Die **wiederh**-Funktion benötigt zwei Argumente. Das erste ist eine Zeichenfolge (Text) bzw. ein Verweis auf eine Zelle, die Text enthält, das zweite ist numerisch. Die Funktion erzeugt dann entsprechend viele Wiederholungen des ersten Zeichens im Text. In unserem Beispiel unterstreicht sie den Titel mit Gleichheitszeichen, wobei sich die Länge aus der Länge des Eintrags in C1 ergibt. Falls Sie zu einem späteren Zeitpunkt einen anderen Titel wählen, brauchen Sie die Formel in C2 nicht etwa zu ändern.

[A4] `zle=monat(sp()-1)` {Spalten B bis M}
[A5] `zle=wiederh("-",breite()+1)` {Spalten A bis M}

Diese Eintragungen über mehrere Zeilen erzeugen die Monatsnamen und einen Längsstrich über das benutzte Arbeitsblatt. Die Funktion **breite()** liefert die Breite einer Spalte, ausgedrückt in Zeichenpositionen. Aus diesem Grund kann sie auch verwendet werden, um eine Linie durch Spalten ungleicher Breite zu ziehen. Die einzelnen Spalten sind jeweils durch ein Leerzeichen getrennt – der Grund für das "+ 1".

[A6] `"VERKAUF`
[B6] `4000`
[C6] `zle=Ver.Jan*1.02` {Spalten C bis M}

Mit diesen Eingaben füllen Sie die Verkaufszahlen für das ganze Jahr aus, ausgehend von der Annahme, daß der Wert im Januar 4000 betrug und daß monatlich ein Zuwachs von 2% erzielt wird.

```
            A        |    B    |    C    |    D    |    E
 1 |                          CASH-FLOW
 2 |                          =========
 3 |
 4 |                     Januar    Februar    März      April
 5 |----------------------------------------------------------
 6 | VERKAUF              4000.00   4080.00   4161.60   4244.83
 7 | KOSTEN               2750.00   2790.00   2830.80   2872.42
 8 |----------------------------------------------------------
 9 | BRUTTOGEWINN         1250.00   1290.00   1330.80   1372.41
10 |
11 | AUFWAND
12 |         Gehälter      700.00    700.00    700.00    700.00
13 |          Werbung      100.00    100.00    100.00    100.00
14 |            Miete      200.00    200.00    200.00    200.00
15 |            Strom       50.00     50.00     50.00     50.00
16 |   Abschreibungen       90.00     90.00     90.00     90.00
17 |----------------------------------------------------------
18 | GESAMTAUFWAND        1140.00   1140.00   1140.00   1140.00
19 |----------------------------------------------------------
20 | NETTOGEWINN           110.00    150.00    190.80    232.42
21 |==========================================================
```

Abbildung 5.1 Die fertige Tabelle (Fenster zeigt die ersten 5 Spalten)

```
[A7] "KOSTEN
     Kos=Ver*0.5+750       {Spalten B bis M}
```

Die Kosten werden mit der Hälfte des Verkaufspreises plus einem Fixkostenbetrag von 750.00 angesetzt.

```
[A8] zle=a5          {Spalten A bis M}
[A9] "BRUTTOGEWINN
     bru=ver-kos       {Spalten B bis M}
```

Dies zieht eine weitere Querlinie und berechnet die monatlichen Bruttogewinnzahlen.

```
[A11] "AUFWAND
[A12] "Gehälter
      zle=700       {Spalten B bis M}
[A13] "Werbung
      zle=100       {Spalten B bis M}
[A14] "Miete
      zle=200       {Spalten B bis M}
[A15] "Strom
      zle=50       {Spalten B bis M}
[A16] "Abschreibungen
      zle=90       {Spalten B bis M}
```

Diese Eingaben betreffen die Aufwendungen, die über das ganze Jahr konstant bleiben. Selbstverständlich können Sie nach Bedarf die Aufwandposten und die dazugehörigen Werte ändern und zusätzliche Posten einfügen bzw. welche auslassen. Nur müssen Sie diese Änderungen dann im weiteren Verlauf des Beispiels (bei Bereichsverweisen) berücksichtigen. Selbst wenn Sie für jeden Monat verschiedene Werte eintragen wollen, ist es im allgemeinen schneller, die Tabelle erst einmal in dieser Weise aufzubauen und die Änderungen später vorzunehmen.

```
[A17] zle = a5       {Spalten A bis M}
[A18] "GESAMTAUFWAND
[B18] zle=summe(sp)       {Zeilen 12 bis 16, Spalten B bis M}
[A19] zle=a5       {Spalten A bis M}
```

Damit haben Sie den gesamten Aufwand für jeden Monat.

Die Funktion **summe()** addiert die Inhalte aller numerischen Felder ihres Arguments. Leere Felder und solche mit alphanumerischem Inhalt bleiben unberücksichtigt. Der Bereichsverweis kann ausdrücklich gegeben werden, etwa als B12:B16. Da jedoch im vorliegenden Beispiel jeder Bereich nur eine einzige Spalte umfaßt, wurde der Einfachheit halber der Bereichsbezeichner "sp" benutzt. Sie brauchen lediglich die Vorschläge von Abacus mit **ENTER** zu akzeptieren bzw. zu überschreiben.

Beachten Sie die zwei unterschiedlichen Anwendungen der Bereichsbezeichner **sp()** und **zle()** in dieser Formel. Zuerst wird **zle()** zur Spezifikation des Bereichs angegeben, in dessen Felder die Formel zu plazieren ist. **sp()** bezeichnet dann den Bereich, dessen Felder zu addieren sind. Die Anfangs– und Endpunkte dieser Bereiche müssen Sie akzeptieren bzw. überschreiben. In diesem Fall behandelt Abacus zunächst den Bereich für die **summen**-Funktion.

```
|A20| "NETTOGEWINN
      net=bru-ges        {Spalten B bis M}
|A21| zle= wiederh("=",breite()+1)     {Spalten A bis M}
```

Die Berechnung des Nettogewinns als Differenz zwischen dem Bruttogewinn und dem Gesamtaufwand vervollständigt unsere Kalkulationstabelle. Nun wollen wir noch einige optische Verschönerungen daran vornehmen. Die Befehle, die wir dazu benötigen, gehören zum Befehlsmenü, das über **F3** erreicht wird.

Als erstes wollen wir die Spaltenbreite ändern. Zu diesem Zweck fordern Sie mit **R** den **raster**-Befehl an, der über ein eigenes Untermenü verfügt. Wählen Sie dort **S** für Spaltenweite:

```
Raster>Spaltenweite,15 VON a BIS a
```

Dann ändern wir die Ausrichtung (Bündigkeit) und das numerische Anzeigeformat einiger Felder:

```
Bündig,besetzt,Text,rechts,Bereich A4:M4
Bündig,besetzt,Text,rechts,Bereich A12:A16
Zahlen,besetzt,Dezimal,Dezimalstellen 2,Bereich A1:M21
```

Für die Zahlen haben wir das Dezimalformat mit 2 Nachkommastellen gewählt. Auf Wunsch können Sie sich ein M davor anzeigen lassen. Der entsprechende Befehl lautet:

```
Zahlen,besetzt,Währung,Minuszeichen,Bereich A1:M21
```

Mit dem **format**–Befehl müssen Sie das Währungssymbol in M ändern.Das Verändern einzelner Zahlen ist denkbar einfach. Angenommen, die Werbekosten im Februar sind höher als erwartet und sollen geändert werden. Drücken Sie **F5** (Sprung auf Feld – "Nach") und geben Sie die Feldadresse ein:

```
Feb.Wer
```

Sofort springt der Cursor in das gewünschte Feld, wo Sie den neuen Wert eintragen.

Sie erinnern sich, daß die Verkaufszahlen anhand einer Formel eingesetzt wurden, welche einen 2%igen Zuwachs pro Monat annahm. Wenn Sie einen anderen Wert in eines dieser Felder einsetzen, geht die zugrundeliegende Formel für dieses einzelne Feld verloren – nicht jedoch für die anderen Felder in der Zeile, welche nach wie vor monatlich 2% Zuwachs aufweisen, wobei der neue Wert mitberücksichtigt wird.

MULTIPLIKATIONS-TABELLEN

Unser nächstes Beispiel könnte sich als nützlich erweisen für ein Kind, das die Multiplikationstabellen lernen muß. Es kann eine bestimmte Tabelle anfordern, die dann am Bildschirm angezeigt wird.

Die Tabelle kann mit dem Neukalkulations-Befehl aktiviert werden, also mit

```
[F3] N
```

Abacus fordert dann die Eingabe einer Zahl an und präsentiert die entsprechende Multiplikationstabelle.

Abbildung 5.2 zeigt ein typisches Beispiel.

Als erstes geben Sie dem Arbeitsblatt eine Überschrift:

```
|B1| "MULTIPLIKATIONSTABELLEN
|B2| wiederh("=",länge(b1))
```

und mit den folgenden drei Eingaben der jeweiligen Multiplikationstabelle einen Titel:

```
[B3] "DIE
[C3] anfn("Welche Tabelle möchtest Du sehen")
[D3] "-ER TABELLE
```

Hier verwenden wir die Funktion **anfn()** zur Anforderung der gewünschten Multiplikationstabelle.

```
              A          | B    | C    | D    | E    | F
 1 |                      MULTIPLIKATIONSTABELLEN
 2 |                      =====================
 3 |                        DIE 7 -ER TABELLE
 4 |                        1 mal     7      =      7
 5 |                        2 mal     7      =     14
 6 |                        3 mal     7      =     21
 7 |                        4 mal     7      =     28
 8 |                        5 mal     7      =     35
 9 |                        6 mal     7      =     42
10 |                        7 mal     7      =     49
11 |                        8 mal     7      =     56
12 |                        9 mal     7      =     63
13 |                       10 mal     7      =     70
14 |                       11 mal     7      =     77
15 |                       12 mal     7      =     84
```

Abbildung 5.2 Eine typische Multiplikationstabelle

Die Funktion **anfn()** akzeptiert als Argument eine Zeichenkette und zeigt diesen Text in der Eingabezeile, gefolgt von einem Fragezeichen, an. Sie wartet dann eine Eingabe mit abschließendem **ENTER** ab. Die eingegebene Zahl erscheint im Feld mit der **anfn()**-Formel.

Während einer normalen Neukalkulation der Tabelle erwartet die Formel keine Eingabe. Die Anzeige der Nachricht und die Aufforderung erfolgen nur beim Eingeben der Formel in das Feld und bei einer erzwungenen Neukalkulation mit dem **neukalk**-Befehl. Ein einmal eingegebener Wert wird im Feld aufbewahrt, bis wieder eine Neukalkulation angefordert wird.

Die restlichen Tabelleneinträge verwenden die Funktion zum Füllen von Spalten und erzeugen auf diese Weise die eigentliche Multiplikationstabelle.

[B4] `sp=kette(zle()-3,2,0)+" *"` {Zeilen 4 bis 15}

Dies ist die komplizierteste Formel im ganzen Beispiel. Sie dient zur Ausgabe des Multiplikators auf jeder Zeile. Dabei wird die Zahl in eine Zeichenkette umgewandelt, damit sie zusammen mit dem Multiplikationszeichen (*) in derselben Zelle angezeigt werden kann.

Die Funktion **kette()** verwandelt eine Zahl in die entsprechende Zeichenkette. Sie übernimmt drei Werte: die Ausgangszahl, einen Code für das Anzeigeformat (0 = dezimal, 1 = exponential, 2 = ganzzahlig, 3 = allgemein) und die Anzahl der Dezimalstellen. Im vorliegenden Fall wird der Wert als ganze Zahl ausgegeben.

Der Wert ergibt sich aus dem Ausdruck **"zle()−3"**, dessen Wert in Zeile vier 1, in Zeile fünf 2 ... und in Zeile fünfzehn 12 ist. Die folgende Ziffer (2) bestimmt die Anzeige als ganze Zahl, und die dritte Ziffer legt im allgemeinen die Anzahl der Dezimalstellen fest. Obwohl diese Angabe natürlich bei ganzen Zahlen irrelevant ist, muß sie dennoch gemacht werden. Wir haben hier eine 0 eingesetzt; es könnte jedoch auch eine andere Zahl verwendet werden.

Schließlich wird das Ergebnis mit dem Multiplikationszeichen (*) *verknüpft* (beim Aneinanderhängen von Ketten spricht man im allgemeinen von Verknüpfen), so daß beide zusammen in einem Feld zur Anzeige gelangen.

[C4] `sp=$c3` {Zeilen 4 bis 15}

Spalte C enthält Kopien des Wertes, der als Antwort auf die **anfn()**-Funktion eingegeben wurde. Die Feldadresse C3 ist mit einem vorangehenden Dollarzeichen als *absolute Feldadresse* gekennzeichnet. Fahren Sie nach Eingabe dieser Formel mit dem Cursor in der Spalte C auf und ab und überprüfen Sie den Inhalt der einzelnen Felder. Dabei werden Sie feststellen, daß alle einen Verweis auf $C3 enthalten, d.h. der Verweis bleibt

gleich unabhängig vom jeweiligen Bezugsfeld. Ein absoluter Feldverweis bezeichnet immer ein ganz bestimmtes Feld in der Tabelle, und zwar aus jeder beliebigen Position ohne Rücksicht auf relative Verhältnisse. Durch Hinzufügen des Dollarzeichens kann jeder Feldverweis absolut gemacht werden.

[D4] `sp="="` {Zeilen 4 bis 15}
[E4] `sp=$C3*(zle()-3)` {Zeilen 4 bis 15}

Diese letzten beiden Spalteneintragungen bedürfen eigentlich keiner Erklärung. Sie erzeugen die Gleichheitszeichen und die Ergebnisse (Produkte) in jeder Zeile. Die letzte Formel multipliziert den Wert der **anfn()**-Funktion in Feld C3 (auch hier wieder eine absolute Feldadresse) mit dem Ausdruck **zle()-3**, der, wie wir bereits gesehen haben, in Zeile vier den Wert 1, in Zeile fünf den Wert 2.... und in Zeile fünfzehn den Wert 12 ergibt.

Unter Zuhilfenahme einiger Befehle machen wir die Tabelle etwas übersichtlicher:

```
Bündig,besetzt,Text,rechts,Bereich B3:B15
Bündig,besetzt,Text,rechts,Bereich D4:D15
Bündig,besetzt,Zahlen,Mittig,Bereich C3
Raster>Spaltenweite,5 VON B BIS B
Raster>Spaltenweite,3 VON C BIS C
Raster>Spaltenweite,2 VON D BIS D
Raster>Spaltenweite,4 VON E BIS E
```

Zur Verwendung der Multiplikationstabelle wird mit dem Befehl **neukalk** eine Neukalkulation angefordert. Dabei erscheint der Text der **anfn()**-Funktion in der Eingabezeile und erwartet eine Eingabe.

SCHECKHEFT-KONTROLLE

Unser nächstes Tabellenbeispiel hilft Ihnen, die Übersicht über Ihr Scheckkonto zu bewahren. Jedesmal, wenn Sie einen Scheck ausstellen, tragen Sie die Details in die vorgesehenen Felder ein, und am Monatsende geben Sie Ihr Gehalt, die Daueraufträge usw. ein. Auf diese Weise haben Sie eine Übersicht über Ein– und Ausgänge und können den Saldo mit dem Kontoauszug der Bank vergleichen.

```
                A            |     B      |      C        |      D
  1 |                   SCHECKKONTO-AUFSTELLUNG
  2 |                   =======================
  3 |
  4 |                    Monat          Januar
  5 |
  6 | Saldovortrag                   2000
  7 | Gehalt                      5273.50
  8 | Verschied. Eink.                  0
  9 |
 10 |                    HABEN                        7273.50
 11 |                                                 ======
 12 |
 13 | Daueraufträge                                      1300
 14 | Gebühren                                              0
 15 |
 16 | Schecks      Datum      Scheck-Nr.               Betrag
 17 |            3/01/85       123456                   500
 18 |           10/01/85       123457                   500
 19 |           14/01/85       123458                322.10
 20 |           17/01/85       123459                   500
 21 |           24/01/85       123460                   500
 22 |           31/01/85       123461                   500
 23 |                ---          ---                   ---
 24 |                ---          ---                   ---
 25 |                ---          ---                   ---
 26 |                ---          ---                   ---
 27 |
 28 |                    SOLL                         4122.10
 29 |                                                 ======
 30 | Saldo                       3151.40
 31 |                             ======
```

Abbildung 5.3 Scheckkonto-Aufstellung

Das Ergebnis sehen Sie in Abbildung 5.3 (mit ein paar zusätzlichen Zahlen).

```
[B1] "SCHECKKONTO-AUFSTELLUNG
[B2] wiederh("=",länge(B1))

[C4] "Monat
[D4] anft ("Monat eingeben")
```

Die Funktion **anft()** ist identisch mit **anfn()**, nur daß sie als Eingabe Text, und nicht Zahlen, erwartet. Bei Verwendung des Neukalkulations-Befehls zeigt Abacus in der Eingabezeile den betreffenden Text an und wartet auf eine Texteingabe, nämlich den Namen des Monats, für den die Aufstellung gemacht werden soll.

```
[A6] "Saldovortrag
[A7] "Gehalt
[A8] "Verschied. Eink.
[C6] anfn(A6+" für "+$D4)
```

Die Anfrage für **anfn()** wird aus dem Text anderer Felder zusammengesetzt, wobei auf relative und auf absolute Feldadressen verwiesen wird.

Wir verwenden nun den **echo**-Befehl zum Kopieren der Formel aus Feld C6 in die Felder C7 und C8. Statt des Bereichsverweises C7:C8 können wir den Bezeichner **sp()** benutzen.

```
Echo,Feld C6,über Bereich sp        {Zeilen 7 bis 8}

[B10] "HABEN
[C10] summe(sp)        {Zeilen 6 bis 8}
```

C10 ist das Summenfeld für alle in dem Monat eingegangenen Gutschriften. Es hat den Namen "haben.monat".

Sein Inhalt wird mit der Summenfunktion berechnet, die wir bereits im ersten Beispiel dieses Kapitels verwendet haben. **summe()** addiert alle numerischen Inhalte der Zellen innerhalb des Bereichs, den ihr Argument angibt. Leere und Textzellen werden nicht berücksichtigt.

Hier hat die Funktion ebenfalls wieder die Form **summe(sp)**, d.h. die Felder, die aufaddiert werden sollen, befinden sich alle in der aktuellen Spalte. Wie üblich fragt Abacus nach dem exakten Anfangs- und Endpunkt und macht einen Vorschlag, der sich auf die bereits gefüllten Tabellenfelder stützt.

```
[C11] wiederh("=",länge(kette(hab.monat,0,2)))
```

Feld C11 unterstreicht das Gesamtguthaben mit Hilfe der inzwischen wohlbekannten Funktionen **wiederh()** und **länge()**. In diesem Fall ist jedoch die exakte Länge des zu unterstreichenden Wertes nicht voraussehbar. Folglich muß die Zahl mit der Funktion **kette()** in eine Zeichenkette umgewandelt werden, unter der Annahme, daß das Dezimalformat mit zwei Dezimalstellen verwendet wird. Die Länge dieser Kette liefert die exakte Anzahl von Zeichen, die unterstrichen werden sollen.

```
[A13] "Daueraufträge
[A14] "Gebühren
[D13] anfn(A13+" für "+$D4)
[D14] anfn(A14+" für "+$D4)
```

Damit können die Belastungen aufgrund von Anfragen mit **anfn()** eingegeben werden, analog zu der oben beschriebenen Methode.

```
[A16] "Schecks
[B16] "Datum
[C16] "Scheck-Nr.
[D16] "Betrag
[B17] zle="---"        {Spalten B bis D}
```

Diese Felder definieren den Tabellenblock, in den Sie dann die Details zu Ihren Schecks eintragen.

```
[B28] "SOLL
[D28] summe(sp)        {Zeilen 13 bis 26}
```

Damit wird die Summe der Sollbeträge (der ausgehenden Schecks) berechnet. Zur Erinnerung: **summe()** addiert nur die numerischen Werte des spezifizierten Bereichs. Leerzellen und Textzellen bleiben unberücksichtigt. Bei der Addition werden alle nicht ausgefüllten Felder in der Scheckliste ausgelassen, und natürlich auch die Überschrift in Spalte D.

```
[A30] "Saldo
[C30] haben.monat-soll.Betrag
```

Mit der Berechnung des Saldos ist unser Arbeitsblatt, was Eingaben anbelangt, abgeschlossen. Was zu tun bleibt, sind optische Verschönerungen.

Echo kann zum Ausfüllen der restlichen Schecktabelle und zum Unterstreichen der Gesamtbeträge verwendet werden. Dieser Befehl kopiert den Inhalt einer einzelnen Zelle in alle Felder eines Bereichs. Der erste der folgenden drei Befehle kopiert z .B. den Inhalt von B17 in alle Felder in dem rechteckigen Block, der durch die beiden Endfelder B18 und D26 abgesteckt wird.

```
Echo,Feld B17,über Bereich B18:D26
Echo,Feld C11,über Bereich D29:D29
Echo,Feld C11,über Bereich C31:C31
```

Danach sollten wir die numerische Anzeige auf Dezimalformat mit 2 Stellen setzen, und zwar für die ganze Tabelle, mit Ausnahme der Scheck-Nummern, die als ganze Zahlen erscheinen sollen.

```
Zahlen,besetzt,Dezimal,Dezimalstellen 2,Bereich A1:D30
Zahlen,besetzt,Ganzzahl,Minuszeichen,Bereich C17:C26
```

Schon des öftern wurde darauf hingewiesen, daß leere Felder für Abacus nicht existieren. Eine Änderung des Formats kann sich also nur auf besetzte Felder beziehen und würde in unserem Beispiel die noch nicht mit Beträgen gefüllten Felder der Schecktabelle nicht betreffen. Dem kann abgeholfen werden, indem man entweder die noch unbesetzten Felder mit "---" füllt oder indem man mit **L** für Leere Felder die Standardvorgabe ändert.

Um den optischen Eindruck zu verbessern, kann die Anordnung der Texte (und der Unterstreichungen) noch von links– auf rechtsbündig geändert werden:

```
Bündig,besetzt,Text,rechts,Bereich B16:D26
Bündig,besetzt,Text,rechts,Bereich C11
Bündig,besetzt,Text,rechts,Bereich D29
Bündig,besetzt,Text,rechts,Bereich C31
```

Die Dimensionen unserer Schecktabelle sind größer als das Bildschirmfenster, so daß wir nicht alles auf einen Blick erfassen können. Um die Endergebnisse zusammen mit den Eingaben über die beiden Funktionen **anft()** und **anfn()** zu sehen, kann der Befehl **öffnung** zum Teilen des Fensters verwendet werden. Die Teilung kann in horizontaler oder in vertikaler Richtung erfolgen, wobei die Trennlinie durch die Cursorposition bestimmt wird.

Eine vertikale Teilung eignet sich für die vorliegende Tabelle sicher am besten. Verlagern Sie den Cursor ins Zentrum des Fensters und geben Sie dann diesen Befehl ein:

```
Öffnung,lotrecht,separat
```

Der Cursor kann mit **F4** von einem Fenster zum andern springen. Im vorliegenden Beispiel sollte mit Hilfe des Cursors die Anordnung so erfolgen, daß das linke Fenster in der oberen Ecke das Feld A1 und das rechte Fenster das Feld B16 anzeigen.

STANDARD-ABWEICHUNG

Unsere nächste Beispieltabelle berechnet die Standard– oder mittlere quadratische Abweichung einer Serie von Zahlen. Sie macht Gebrauch von den Abacus-Namenfunktionen, so daß die verwendeten Formeln zumeist ohne Erklärung sofort einsichtig sind.

Die Tabelle arbeitet mit einem Layout, welches eine Berechnung nach Spalten statt der allgemein üblichen zeilenweisen Berechnung verlangt.

Normalerweise gilt die Regel, daß in einer Formel nur Verweise auf Felder vorkommen dürfen, die oberhalb oder links des betreffenden Feldes liegen (einschließlich der Zeile und der Spalte, welche die Koordinaten des Feldes bilden).

Bei Nichteinhaltung dieser Regel, wie etwa im vorliegenden Beispiel, muß mit fehlerhaften Resultaten gerechnet werden. Allerdings läßt sich in den meisten Fällen ein richtiges Ergebnis erzielen, indem man eine Neukalkulation des Arbeitsblattes mit **neukalk** oder, wie hier, mit spaltenweiser Berechnung, erzwingt.

```
[B1] "STANDARDABWEICHUNG
[B2] wiederh("=",länge(B1))
[B4] "Wert
```

```
[C4] "Abweichung
[D4] "Quadrat
[B5] sp=zle()        {Spalten 5 bis 14}
```

```
             A        |    B     |    C     |    D     |    E
  1 |               STANDARDABWEICHUNG
  2 |               ==================
  3 |
  4 |               Wert          Abweichung Quadrat
  5 |                    5.00         -4.50     20.25
  6 |                    6.00         -3.50     12.25
  7 |                    7.00         -2.50      6.25
  8 |                    8.00         -1.50      2.25
  9 |                    9.00         -0.50      0.25
 10 |                   10.00          0.50      0.25
 11 |                   11.00          1.50      2.25
 12 |                   12.00          2.50      6.25
 13 |                   13.00          3.50     12.25
 14 |                   14.00          4.50     20.25
 15 |
 16 |    Mittelwert      9.50     Varianz        8.25
 17 |                             Std.Abw        2.87
 18 |                                        ---------
```

Abbildung 5.4 Berechnung der Standardabweichung

Die letzte Formel setzt zu Testzwecken eine Reihe von Werten in die Felder der Spalte B. Nach Vervollständigung der Tabelleneintragungen können diese Werte durch effektive Zahlen ersetzt werden. Die hier beschriebene Tabelle bewältigt nicht mehr als zehn Werte, doch können Sie das bei Bedarf leicht ändern.

```
[A16] "Mittelwert
[B16] durchschn(wert)        {Zeilen 5 bis 14}

abweichung=wert-$mittel.wert      {Zeilen 5 bis 14}
quadrat=abw*abw       {Zeilen 5 bis 14}

[C16] "Varianz
[D16] durchschn(quadrat)       {Zeilen 5 bis 14}
```

Diese Formeln zeigen, daß die Varianz einer Serie von Zahlen als der Durchschnitt des Quadrats der Abweichungen vom Mittelwert definiert ist,

```
[C17] "Std.Abw
[D17] qwurzel(Varianz)       {von D bis D}
```

und daß die Standardabweichung errechnet werden kann als die Quadratwurzel der Varianz.

```
[D18] wiederh("- ",länge(kette(std.qu,3,0)))
```

Die Zahlen in diesem Beispiel werden im Allgemeinen Format belassen, um den ganzen Zahlenbereich abzudecken. Die Unterstreichung verwendet die Länge der Textkette in Übereinstimmung mit der im Allgemeinen Format angezeigten Zahl im darüberliegenden Feld (mit der Feldadresse "std.qu").

Das Erscheinungsbild der Tabelle verbessert sich, wenn Sie für den Text im Bereich B4:D4 mittige und für die Zahlen im Bereich B16:D17 linksbündige Anordnung vorsehen.

Beim Versuch, dieses Beispiel mit neuen Werten in den Feldern der B-Spalte zu betreiben, werden Sie bemerken, daß die Ergebnisse nicht richtig ausfallen. Die Ursache liegt darin, daß die Neukalkulation des Arbeitsblattes Zeile um Zeile, von oben nach unten, durchgeführt wird. Jede Änderung wird folglich auf der Basis eines falschen Mittelwertes berechnet, da der neue Mittelwert erst *nach* den Abweichungen kalkuliert wird. Als Lösung für dieses Problem bietet sich eine andere Methode der Kalkulation an, welche spaltenweise von links nach rechts vorgeht. Sie können diese Art der Kalkulation mit dem **format**-Befehl wählen.

Aus dem Format-Menü Auswahlzeile **B** (Berechnungsreihenfolge) drücken, welche sofort von ZLE auf SP wechselt. Den Befehl mit **ENTER** verlassen.

Wenn Sie jetzt einen Wert in der B-Spalte neu eintragen, erhalten Sie ein korrektes Resultat, weil der neue Mittelwert vor den Abweichungen berechnet wird. Die Möglichkeit, eine solche Umstellung der Kalkulation vorzunehmen, ist zwar sehr praktisch, sollte aber dennoch nur in Ausnahmefällen verwendet werden, weil die spaltenweise Berechnung erheblich zeitaufwendiger ist als die zeilenweise.

Die aktuellen **format**-Einstellungen werden mit dem Arbeitsblatt zusammen auf Microdrive-Kassette abgespeichert und auch jederzeit wieder in den Arbeitsspeicher miteingelesen, wenn die Tabelle geladen wird.

EIN HAUSHALTSBUDGET

Nun zu einem Beispiel, welches Ihnen beim Planen und Verwalten Ihres Haushaltsbudgets hilft. Ihre Schätzungen tragen Sie unter einer Anzahl von Rubriken für jedes Vierteljahr ein. Sie erhalten dann automatisch die vierteljährlichen Gesamtausgaben, das Jahrestotal und die durchschnittlichen monatlichen Kosten.

Warten Sie mit dem Eintragen der Zahlen, bis Sie das Tabellengerüst fertiggestellt haben. Auf diese Weise können Sie das Format der numerischen Werte mit der **L**(Leerfelder-) Option des **zahlen**-Befehls ändern, die wir etwas später beschreiben.

```
    A|      B       |C|   D    |E|   F    |G|   H    |I|   J    |K
 1 |                    HAUSHALTSBUDGET
 2 |                    ===============
 3 |
 4 | =========================================================
 5 | !              !          GESCHÄTZTE AUSGABEN           !
 6 | ! Posten       ! Jan-Mär ! Apr-Jun ! Jul-Sep ! Okt-Dez !
 7 | =========================================================
 8 | !Miete         ! 4000.00 ! 4000.00 ! 4000.00 ! 4000.00 !
 9 | !Steuern       !         ! 4500.00 !         !         !
10 | !Gas           ! 1500.00 !  800.00 !  600.00 ! 1500.00 !
11 | !Strom         !  400.00 !  300.00 !  300.00 !  400.00 !
12 | !Wasser        !         !  350.00 !         !  350.00 !
13 | !Telefon       ! 1500.00 ! 1500.00 ! 1500.00 ! 1500.00 !
14 | !Versicherung  !         !         !         !         !
15 | !Kleider       !         !         !         !         !
16 | !Lebensmittel  !         !         !         !         !
17 | !Auto          !         !         !         !         !
18 | !Zeitungen     !         !         !         !         !
19 | !Freizeit      !         !         !         !         !
20 | !Ersparnisse   !         !         !         !         !
21 | =========================================================
22 |  Viertelj.-Total ! 7400.00 !11450.00 ! 6400.00 ! 7750.00 !
23 |                 =========================================
24 |
25 |                 Jährliche Monatliche
26 |
27 |      Zahlungen  M33000.00  M2750.00
28 |                 ========= =========
```

Abbildung 5.5 Beispiel eines Haushaltsbudgets

```
[D1] "HAUSHALTSBUDGET
[D2] wiederh("=",länge(D1))
```

Jetzt wird erst die Struktur der Tabelle mit den übersichtlich abgetrennten Spalten aufgebaut:

```
[A4] zle=wiederh("-",breite()+1)      {Spalten A bis K}
[A5] sp="!"      {Spalten 5 bis 20}
```

Und hier die Liste aller Befehle zur Vervollständigung der Struktur:

```
Raster>Spaltenbreite,16 VON B BIS B
Raster>Spaltenbreite,8 VON D BIS J
Raster>Spaltenbreite,1 VON A BIS A
Raster>Spaltenbreite,1 VON C BIS C
Raster>Spaltenbreite,1 VON E BIS E
```

```
Raster>Spaltenbreite,1 VON G BIS G
Raster>Spaltenbreite,1 VON I BIS I
Raster>Spaltenbreite,1 VON K BIS K

Echo,Feld A5,über Bereich C5:C22
Echo,Feld A5,über Bereich E6:E22
Echo,Feld A5,über Bereich G6:G22
Echo,Feld A5,über Bereich I6:I22
Echo,Feld A5,über Bereich K5:K22
Echo,Feld A4,über Bereich B7:J7
Echo,Feld A4,über Bereich B21:K21
Echo,Feld A4,über Bereich C23:K23

[A7] "!-
[F5] "GESCHÄTZTE AUSGABEN
[B6] "Posten
[D6] "Jan-Mär
[F6] "Apr-Jun
[H6] "Jul-Sep
[J6] "Okt-Dez

[B8] "Miete
[B9] "Steuern
[B10] "Gas
[B11] "Strom
[B12] "Wasser
[B13] "Telefon
[B14] "Versicherung
[B15] "Kleider
[B16] "Lebensmittel
[B17] "Auto
[B18] "Zeitungen
[B19] "Freizeit
[B20] "Ersparnisse

[B22] "Viertelj.-Total

[D22] summe(sp)     {Zeilen 8 bis 20}
[F22] summe(sp)     {Zeilen 8 bis 20}
[H22] summe(sp)     {Zeilen 8 bis 20}
[J22] summe(sp)     {Zeilen 8 bis 20}

[D25] "Jährliche
[F25] "Monatliche
[B27] "Zahlungen

[D27] summe(D22:J22)
[F27] jährl.zah/12
[D28] wiederh("=",länge(kette(jährl.zah,0,2))+1)
[F28] D28
```

Beachten Sie, daß die Unterstreichung der letzten beiden Zahlen ein "Währungsformat" annimmt; folglich ist die Länge des Unterstreichungsstrichs für eine Zahl mit zwei Dezimalstellen und zusätzlich eine Stelle für M gerechnet.

Für eine bessere Präsentation nehmen wir noch ein paar Schönheitskorrekturen vor, z .B. um den Text im Bereich B22:B27 rechtsbündig (Viertelj.-Total und Zahlungen) und die Beträge in den Feldern für jährliche und monatliche Zahlungen linksbündig anzuordnen.

Außerdem ist nun noch das numerische Anzeigeformat zu ändern. Die meisten der für numerische Eingabe vorgesehenen Felder sind ja noch leer – mit Absicht, weil wir jetzt die Standardformatvorgaben ändern können, die dann für die gesamte Haushaltsbudget-Tabelle gelten.

Der folgende Befehl setzt das Standardformat auf "Währung":

```
Zahlen,leer,Währung,Minuszeichen
```

Die Tabelle in Abbildung 5.5 verwendet Dezimalformat mit zwei Stellen, mit Ausnahme der jährlichen und monatlichen Zahlungen, welche im Währungsformat angezeigt werden. Die entsprechenden Befehle lauten:

```
Zahlen,leer,Dezimal,Dezimalstellen 2
Zahlen,besetzt,Währung,Minuszeichen,Bereich D27:F27
```

(Der letzte Befehl bezieht sich auf bereits besetzte Felder.)

Mit dem **format**-Befehl setzen Sie das Währungssymbol auf M.

Nun können Sie beliebige Eintragungen in Ihrem Haushaltsbudget vornehmen, wobei Sie den Cursor mit den Pfeiltasten oder mit **F5** ("Nach") und Eingabe der Feldadresse, etwa

```
apr.gas
```

steuern.

BALKENDIAGRAMM MIT AUTOSKALA

Das Diagramm stellt 12 Werte grafisch dar, beschriftet nach Monaten. Eingelesen werden die Werte aus 12 Feldern über dem Balkendiagramm, wobei die vertikale Skala sich automatisch so verändert, daß alle Werte angezeigt werden können. Die Darstellung eignet sich nur für positive Werte.

Als erstes sollte die Spaltenbreite in Spalte A auf 5, in Spalte B auf 1 und auf 3 in den Spalten C bis N eingestellt werden. Für diesen Zweck im Menü des **raster**-Befehls die Spaltenbreite entsprechend wählen.

[**C2**] `zle=0` {Spalten C bis N}

```
      A    |B| C  | D | E | F | G | H | I | J | K | L | M | N
 1 |
 2 |         3   4   3   2   3   4   3   4   1   2   3   2
 3 |                     SKALIERTES BALKENDIAGRAMM
 4 |    5 !
 5 |  4.5 !
 6 |    4 !     ***             ***     ***
 7 |  3.5 !     ***             ***     ***
 8 |    3 ! *** *** ***     *** *** *** ***
 9 |  2.5 ! *** *** ***     *** *** *** ***             ***
10 |    2 ! *** *** *** *** *** *** *** ***     *** *** ***
11 |  1.5 ! *** *** *** *** *** *** *** ***     *** *** ***
12 |    1 ! *** *** *** *** *** *** *** *** *** *** *** ***
13 |  0.5 ! *** *** *** *** *** *** *** *** *** *** *** ***
14 |    0 ! -----------------------------------------------
15 |        Jan Feb Mär Apr Mai Jun Jul Aug Sep Okt Nov Dez
```

Abbildung 5.6 Balkendiagramm mit veränderlicher Skala

Zeile 2 wird später die Werte enthalten, die grafisch dargestellt werden sollen – im Moment mit Testeingabe gefüllt.

[**F3**] `"SKALIERTES BALKENDIAGRAMM`
[**P2**] `ganzzahl(max(C2:N2)/5+1)*5`
[**Q2**] `ganzzahl(min(C2:N2)/5)*5`

Die Felder P2 und Q2 enthalten die maximalen und die minimalen Werte für die vertikale Diagrammskala. Sie liegen "außer Sichtweite", d.h. sie erscheinen nicht im Fensterausschnitt, der das Balkendiagramm enthält. Ihre Ausgangswerte sind fünf bzw. null.

Die **max()**-Funktion ermittelt das Maximum, d.h. den höchsten numerischen Wert in dem Feldbereich, der ihr Argument ist. Analog dazu ermittelt **min()** den minimalen, d.h. den kleinsten numerischen Wert in einem Bereich.

Sehen wir uns die Formel in Q2 etwas genauer an. **min()** ermittelt den kleinsten Wert im angegebenen Feldbereich und dividiert ihn durch 5. Etwaige Nachkommastellen werden durch die Funktion **ganzzahl()** abgeschnitten. Angenommen, der kleinste Wert ist 13, dann ergibt die Division durch 5 das Ergebnis 2.6, und die Funktion **ganzzahl()** macht daraus 2. Durch Multiplikation mit 5 erhalten wir schließlich den Wert 10 – das größte Vielfache von 5, welches kleiner als der Minimalwert ist.

Ähnlich verhält sich die Formel in P2, nur daß sie den höchsten Wert des Bereichs ermittelt und vor der Multiplikation mit 5 eins dazuzählt. Ausgehend von einem Höchstwert von 21 können Sie nachprüfen, daß die Formel einen Wert von 25 ergibt – das kleinste Vielfache von 5, welches größer als der Maximalwert ist.

Die Werte in diesen beiden Feldern bilden folglich in jedem Fall eine Umklammerung der Werte in den Feldern C2 bis N2; ihre Differenz ist immer ein Vielfaches von fünf.

Unsere nächste Formel zeigt die vertikale Diagrammskala in der A-Spalte an:

[A4] `sp=$Q2+(14-zle())*($P2-$Q2)/10` {Zeilen 4 bis 14}

Der Abstand zwischen aufeinanderfolgenden Zahlen auf der Skala ist (P2-Q2) / 10. Da die Differenz zwischen dem Inhalt von P2 und Q2 ein Vielfaches von 5 ist, ergibt sich immer ein einfacher Wert als Skalenabstand.

Dieser Abstand wird mit einer Zahl **(14–zle())** multipliziert, die in Zeile 14 bei Null beginnt und in Einerschritten bis zu einem Wert von 10 in Zeile 4 zunimmt. Das Resultat wird zum minimalen Wert aus Q2 addiert und produziert so die einzelnen Feldinhalte.

Das Endergebnis sieht dann so aus, daß der Wert aus Feld Q2 in A14 und der aus P2 in A4 ausgegeben wird. Diese beiden Felder bilden die obere und die untere Begrenzung der Skala, und die dazwischen liegenden Felder enthalten regelmäßig abgestufte Werte.

[B4] `sp="!"` {Zeilen 4 bis 14}
[B14] `zle=wiederh("-",breite()+1)` {Spalten B bis N}
[C15] `zle=monat(sp()-2) (bis 3)` {Spalten C bis N}

Damit werden die Achsen eingezeichnet, mit Beschriftung für die horizontale Achse (Monatsnamen). Dabei wurde der SuperBASIC ähnliche Aufteilungsoperator (bis) verwendet, um lediglich die ersten drei Buchstaben jedes Monats auszugeben.

[C4] `wenn(index(1,zle()) > index(sp(),2),"","***")`

Diese Formel leistet die ganze Arbeit zur Erstellung der Balken und muß in jedes Feld des Tabellenblocks kopiert werden:

```
Echo,Feld C4,über Bereich C4:N13
```

Die Formel bedarf einer kurzen Erläuterung. Sie verwendet die **wenn()**-Funktion für die Entscheidung, ob ein Balkenstück angezeigt werden soll. **wenn()** übernimmt drei Argumente, wovon das erste ein Ausdruck sein muß, der ein numerisches Resultat liefert. Wenn dieses verschieden von Null ist, zeigt das Feld das zweite Argument, welches alphanumerisch (Text) oder numerisch sein kann. Andernfalls, d.h. wenn das Resultat Null ist, wird das dritte Argument angezeigt, welches ebenfalls eine Textkette oder eine Zahl sein kann.

Die Formel vergleicht in jedem einzelnen Feld die Zahl in Spalte 1 der betreffenden Zeile (der Beschriftung der vertikalen Achse) mit der Zahl in Zeile 2 der betreffenden Spalte (dem grafisch darzustellenden Wert). Wenn die Achsenbeschriftung größer ist als der Grafikausgangswert, ist die Bedingung wahr (1), und es erfolgt keine Anzeige; wenn die Beschriftung einen kleineren oder den gleichen Wert aufweist, ergibt die Bedingung Null, worauf das Feld mit drei Sternchen gefüllt wird. Auf diese Weise wird in jeder Spalte ein Balken in der richtigen Höhe eingezeichnet.

Da für alle Felder genau dieselbe Formel verwendet wird, kann die Feldadresse weder absolut noch relativ sein. Der Verweis auf die grafisch darzustellenden Werte muß sich von Spalte zu Spalte ändern, d.h. muß bezüglich der Spalte relativ sein, sich jedoch bei der zeilenweisen Abwärtsbewegung immer auf Zeile 2 beziehen. Was wir in diesem Fall brauchen, ist eine Art Feldverweis, der relativ in Bezug auf Spalten, jedoch absolut in Bezug auf Zeilen ist.

Zum Glück kann zu diesem Zweck die **index()**-Funktion herhalten, die zwei Parameter übernimmt, eine Spaltenzahl und eine Zeilenzahl, und als Ergebnis den Inhalt des Feldes im Schnittpunkt liefert. Damit lassen sich beliebige Kombinationen von absoluten und relativen Verweisen herstellen, zum Beispiel:

Funktion	Spaltenverweis	Zeilenverweis
index(5,5)	absolut	absolut
index(sp(),5)	relativ	absolut
index(5,zle())	absolut	relativ
index(sp(),zle())	relativ	relativ

Folglich liefert die Funktion **index(sp(),2)** den Inhalt des Feldes in Feld zwei der aktuellen Spalte und **index(1,zle())** den Inhalt der Spalte 1 (A) der aktuellen Zeile.

Experimentieren Sie durch Eingabe verschiedener Werte in den Feldern C2 bis N2 und beobachten Sie, wie sich dies im Balkendiagramm auswirkt.

TILGUNGSPLAN

Nun zu einem Beispiel, welches die monatlichen Rückzahlungen auf eine Hypothek berechnet. Sie werden aufgefordert, die Höhe der Hypothek, den Zinssatz, die Kreditdauer in Jahren und den Monat der ersten Fälligkeit einzutragen. Aufgrund dieser Angaben kalkuliert Abacus die zu leistenden Zahlungen und gibt sie zusammen mit einem vollständigen Tilgungsplan für die gesamte Kreditdauer aus. Aus der Tabelle geht hervor, wie hoch der restliche Kreditbetrag zu Beginn jedes Monats bis zur vollständigen Rückzahlung ist.

Viele Berechnungen in dieser Tabelle arbeiten auf der Basis von Werten, die mit der Funktion **anfn()** angefordert werden.

Berechnung der Rückzahlungen

In diesem Abschnitt bauen wir den Teil des Arbeitsbogens auf, der Ihre Eingabe entgegennimmt und daraus die monatlich fälligen Rückzahlungen berechnet. Nach Eintragen der Formeln und einiger Werte als Antwort auf die **anfn()**-Funktion sollte Ihre Tabelle der Abbildung 5.7 entsprechen.

```
[C1] "HYPOTHEK-RÜCKZAHLUNGS-KALKULATION
[C2] wiederh("=",länge(C1))

[B4] "Kredit
[C4] anfn("Höhe der Hypothek")
```

Die nächsten drei Eintragungen fordern die Eingabe des Zinssatzes an. Diese erfolgt in Feld H4, weit außerhalb des sichtbaren Tabellenfensters, so daß sie normalerweise nicht im Blickfeld liegt. Tragen Sie den Zinssatz als Prozentsatz ein, ohne das Prozentzeichen, also einfach 12 für 12%. Die restlichen Formeln arbeiten mit einem Dezimalbruch, so daß 12% auf 0.12 umgewandelt werden muß. Dies besorgt die Formel in C5.

```
              A    |    B     |    C     |    D     |    E
  1 |                          HYPOTHEK-RÜCKZAHLUNGS-KALKULATION
  2 |                          =================================
  3 |
  4 |               Kredit     25,000.00
  5 |               Zinssatz   14.00%                Monat
  6 |               Dauer      25         Beginn     4
  7 |                                                (April)
  8 |                                     RÜCKZAHLUNGEN
  9 |                                     --------------
 10 |                          jährlich      3637.46
 11 |                          monatlich      303.12
 12 |                                     --------------
```

Abbildung 5.7 Berechnung der Rückzahlungen

```
[H4] anfn("Zinssatz in Prozent")
[B5] "Zinssatz
[C5] H4/100

[B6] "Dauer
[C6] anfn("Kreditdauer in Jahren (max. 35)")

[E5] "Monat
[D6] "Beginn
[E6] anfn("Monat der 1. Zahlung (Jan=1, Feb=2, usw.)")
[E7] '(' + monat(E6) + ")"
```

In dieser letzten Formel setzen wir die Textkonstanten in Apostrophe (Hochkommas). Wenn das erste Zeichen ein Anführungszeichen ist, interpretiert Abacus die nachfolgenden Zeichen als Texteingabe, und nicht als eine Formel, und das soll vermieden werden.

```
[D8] "RÜCKZAHLUNGEN
[D9] wiederh("-",länge(D8))
```

```
[C10] "jährlich
[D10] hyp.kred*hyp.zins/(1-(1+hyp.zins)^(- hyp.dauer))
```

Diese Formel geht bei der Kalkulation der jährlichen Rückzahlung davon aus, daß der Zins einmal im Jahr berechnet und zum ausstehenden Kreditbetrag addiert wird, bevor die zwölf monatlichen Rückzahlungen erfolgen.

```
[C11] "monatlich
[D11] jähr.rück/12
[D12] D9
```

So – mit dieser Tabelle können die Rückzahlungen bereits kalkuliert werden. Verwenden Sie den Befehl **neukalk** zur Eingabe der erforderlichen Informationen.

Auch hier können wir eine Reihe ästhetischer Verbesserungen vornehmen, z .B. das Anzeigeformat mancher Werte anhand des **zahlen**-Befehls ändern. Es lohnt sich bei dieser Anwendung nicht, das Standardformat (leere Felder) zu ändern, weil die einmal erstellte Tabelle ja nicht für weitere Eingaben vorgesehen ist.

```
Zahlen,besetzt,Prozent,Dezimalstellen 2,Bereich C5
Zahlen,besetzt,Währung,Minuszeichen,Bereich C4
Zahlen,besetzt,Währung,Minuszeichen,Bereich D10:D11
```

Außerdem sieht es besser aus, wenn die Zahlen in den Zeilen 4, 5 und 6 linksbündig angeordnet werden:

```
Bündig,besetzt,Zahlen,links,Bereich C4:E6
```

ARBEITSBLATT TILGUNGSPLAN

In diesem Abschnitt beschreiben wir, wie Sie die vorangehende Tabelle durch eine Zusammenstellung der Rückzahlungen ergänzen können. Den ersten Teil einer solchen Übersicht für die Werte aus Abbildung 5.7 sehen Sie in Abbildung 5.8.

```
          A        |    B        |    C      |    D      |    E
15 |                              TILGUNGSPLAN
16 |                              ============
17 |
18 |               Jahr                        1           2           3
19 |                              ------------------------------------
20 |               April          28500.00    28343.30    28164.65
21 |               Mai            28196.88    28040.17    27861.53
22 |               Juni           27893.76    27737.05    27558.41
23 |               Juli           27590.63    27433.93    27255.29
24 |               August         27287.51    27130.81    26952.17
25 |               September      26984.39    26827.69    26649.04
26 |               Oktober        26681.27    26524.57    26345.92
27 |               November       26378.15    26221.44    26042.80
28 |               Dezember       26075.03    25918.32    25739.68
29 |               Januar         25771.90    25615.20    25436.56
30 |               Februar        25468.78    25312.08    25133.44
31 |               März           25165.66    25008.96    24830.31
32 |
33 |               Jahr                        1           2           3
34 |
35 |  Jahresabschluß              24862.54    24705.84    24527.19
```

Tabelle 5.8 Tilgungsplan

```
[C15] "TILGUNGSPLAN
[C16] wiederh("=",länge(C15))
[B18] "Jahr
[C18] zle=sp()-2            {Spalten C bis AK}
[B19] zle=wiederh("-",breite()+1)     {Spalten B bis AK}
[B20] sp=monat(zle()-20+$mon .Beg)     {Spalten 20 bis 31}
```

Damit werden die einzelnen Rubriken mit Überschriften versehen. Was jetzt folgt, sind die Formeln zur Kalkulation der Werte. Fangen wir beim ersten Posten an, dem anfänglichen Kreditbetrag, der sich aus der eigentlichen Hypothek, zusammen mit den aufgelaufenen Zinsen des ersten Jahres, ergibt:

```
[C20] hyp.kre*(1+hyp.zins)
```

Anschließend wird der restliche Teil der ersten Zeile berechnet – durch Subtraktion der jährlichen Rückzahlung und Addition des Zinsbetrags des laufenden Jahres. Natürlich wäre es unsinnig, die Kalkulation über den Ablauf der Kreditperiode hinaus fortzuführen. Um dies zu verhindern, stellen wir anhand der **wenn()**-Funktion sicher, daß eine Null in das Feld plaziert wird, sobald die Jahreszahl (gegeben durch sp()-2) größer ist als die Kreditdauer.

```
[D20] zle=wenn((sp()-2)>$hyp.dau,0,(C20- $jähr.rück)
      *(1+$hyp.zins))          {Spalten D bis AK}
```

Der übrige Teil der Tabelle kann mit einer einzigen Formel gefüllt werden. In das erste Feld geben wir eine Formel ein, welche die montliche Rückzahlung vom Wert des darüberliegenden Feldes subtrahiert. Auch hier sorgt die **wenn()**-Funktion wieder dafür, daß die Kalkulationen nicht über das Jahr der vollständigen Rückzahlung hinausgehen.

```
[C21] wenn((sp()-2)>$hyp.dau,0,C20- $monatl.rück)
```

Mit Hilfe von **echo** können Sie die Formel aus Feld C21 auf den Bereich C21:AK31 kopieren:

```
Echo,Feld C21,über Bereich C21:AK31
```

Um die Aufstellung zu vervollständigen, hängen wir unten eine Zeile mit den noch ausstehenden Kreditbeträgen am Ende jedes Jahres an, und der besseren Übersicht halber setzen wir darüber eine Kopie der Zeile 18 mit den Jahreszahlen.

```
[B33] zle=ja.dau        {Spalten B bis AK}
[A35] "Jahresabschluß
[C35] zle=wenn((sp()-2)>$hyp.dau,0,C31- $monatl.rück)
          {Spalten C bis AK}
```

Die ganze Tabelle, einschließlich der Jahresabschlußziffern, sollte entweder auf das Anzeigeformat "Währung" oder Dezimal mit zwei Dezimalstellen gesetzt werden. Die Bereiche für diese Änderungen sind C20:AK31 und C35:AK35.

FOURIER-ANALYSE

Der französische Gelehrte Fourier zeigte, daß sich eine periodische Schwingung beliebiger Form aus einer Reihe von Sinus– oder Kosinusschwingungen von bestimmter Amplitude und Frequenz zusammensetzt. Der Aufbau komplizierter Schwingungen aus reinen Sinus– und Kosinuswellen ist unter dem Namen Fourier-Synthese bekannt und wird heute beispielsweise bei Synthesizern (Tongeneratoren) verwendet.

Die Umkehrung dieses Vorgangs, die Zerlegung einer Schwingungsbewegung in eine Anzahl reiner Sinus– oder Kosinusschwingungen, nennt man Fourier-Analyse. Unser Beispiel ermöglicht die Durchführung einer Fourier-Analyse an jeder beliebigen Kurve. Sie brauchen nichts weiter zu tun, als die Höhe der Kurve an 16 in regelmäßigem Abstand voneinander liegenden Punkten einzugeben; die ganze Rechenarbeit überlassen Sie den Formeln in der Tabelle. Die Formeln gründen auf der Annahme, daß die Kurvenform sich nach der 16. Teilstrecke wiederholt, d.h. daß der 17. Wert identisch mit dem ersten, der 18. identisch mit dem zweiten ist, usw.

Berechnung der Fourier-Transformation

Es empfiehlt sich, die Auto-Kalkulation bei dieser recht zeitaufwendigen Berechnung auszuschalten. (Über **F3** mit dem **format**-Befehl.)

```
[C1] "FOURIER-ANALYSE
[C2] wiederh("=",länge(C1))

[B3] "Funktion:
[A7] "Eingabe
[A8] "Werte
```

Die Kosinus-Komponenten

Die Eingabewerte kommen in die sechzehn Felder von B9 bis B24.

```
sp=zle()-9      {Zeilen 9 bis 24}
```

Als nächstes die Überschriften für die Tabelle, welche die Kosinus-Komponenten der Kurve berechnet. Das Resultat enthält die Summe der kosinusartigen Teilstrecken in der Eingabe.

```
[E3] "Transformation:
[E4] "Kosinus
```

[D6] `"Zyklen`
`zle=sp()-5` {Spalten E bis T}
[D8] `"Probe`

Vielleicht überrascht es Sie, daß für die ganze Kosinus-Transformation nur eine einzige Formel benötigt wird. Zu diesem Zweck wird in jeder Zeile der Eingabewert mit dem Kosinus eines Winkels (in Bogenmaß) multipliziert, der sich wie folgt berechnet:

Winkel = 2 * pi() * Zeilennummer * Spaltennummer / 16

wobei die Zeilen– und Spaltennummern die Werte aus der Zeile "Zyklen" bzw. der Spalte "Probe" sind. Beide zählen von null bis fünfzehn. Der Divisor am Schluß (16) ist die Anzahl der Teilpunkte der Kurve (Eingabe/Ausgabe).

[E9] `index(2,zle())*kos(pi()*(zle()- 9) *(sp()-5)/8)`

Mit **echo** kann jetzt der Inhalt von E9 auf alle Felder im Bereich E10 bis T24 kopiert werden.

Die Endergebnisse, d.h. die sechzehn Ausgabewerte, kommen durch Aufaddieren der Feldinhalte der einzelnen Spalten zustande:

[A26] `"Komponenten`
[E26] `zle=summe(sp)` {Zeilen 9 bis 24, Spalten E bis T}

Die Sinus-Komponenten

Die Berechnung der Sinus-Komponenten folgt genau dem gleichen Muster, wobei die Ergebnisse die Summe aller sinusartigen Teilstrecken in der Eingabe sind.

[X4] `"Sinus`
[X6] `zle=sp()-24` {Spalten X bis AM}
[X9] `index(2,zle())*sin(pi()*(zle()- 9)*(sp()-24)/8)`
{Spalten X bis AM}

Kopieren Sie jetzt den Inhalt des Feldes X9 mit **echo** über den Bereich von X10 bis AM24, um den übrigen Teil der Tabelle aufzufüllen, und den Inhalt von C9 auf die Spalte V, von V9 bis V24 (kopiert die "Probe"-Werte).

[X26] `zle=summe(sp)` {Zeilen 9 bis 24, Spalten X bis AM}

Das Leistungsspektrum

Abgesehen von reinen Sinus– oder reinen Kosinuskurven produziert jede eingegebene Kurve sowohl Komponenten in der Sinus– als auch in der Kosinus-Transformation. Wenn Sie viele verschiedene Typen von Kurven eingeben, werden manche auch negative Komponenten erzeugen. Um Ergebnisse zu erzielen, die beide Transformationen kombinieren und niemals negative Werte ausgeben, nehmen wir noch eine weitere Kalkulation vor, welche die Quadrate der Sinus– und der Kosinus-Komponenten addiert. Bei einer echten Schwingung drückt das Resultat z.B. die Leistung (Energie) der Schwingung bei jeder Frequenz aus, unabhängig davon, ob es sich um eine Sinus– oder Kosinus– Komponente handelt. Man spricht im allgemeinen vom "Potenz-Spektrum", welches aufzeichnet, wieviel von jeder Frequenz vorhanden ist. Im vorliegenden Fall wollen wir die Quadratwurzel des Potenz-Spektrums berechnen, um es dem Wertebereich unserer einfachen Grafik anzugleichen.

[C28] `"Potenz`
[E28] `zle=qwurzel(kos.Komp*kos.Komp + sin.Komp*sin.Komp)`
{Spalten E bis T}

Grafische Darstellung Der Fourier Transformation

Die Ergebnisse dieser Berechnungen lassen sich grafisch sehr viel anschaulicher darstellen. Für eine hohe Qualität empfehlen wir Ihnen die Verwendung des **Export**-Befehls, der eine Datei erstellt, die dann von EASEL, dem QL-Grafikprogramm, eingelesen und verarbeitet werden kann. Für eine etwas bescheidenere grafische Darstellung der Fourier-Analyse innerhalb von Abacus genügen die folgenden zusätzlichen Schritte.

Die Ausgabe-Grafiken sind nur halb so groß wie die Eingabe-Grafik, da die höchste meßbare Frequenz zahlenmäßig gleich der Hälfte der Eingabepunkte ist. Alle Angaben sind im ersten Teil der Ergebnisse enthalten.

Der erste Teil stellt die Eingabewerte als Säulendiagramm dar.

[A30] `"Grafik`
[A31] `"max=`
[B31] `max(sp)` {Zeilen 9 bis 24}

```
               A       |      B       |      C       |      D
30 | Grafik
31 | max =                 10.00          80.00 = max
32 | min =                -10.00           0.00 = min
33 | ................*        *
34 | ...............*         .................*
35 | .............*           *
36 | ...........*             *
37 | ........*                *
38 | .....*                   *
39 | ..*                      *
40 | *                        *
41 | *
42 | *
43 | ..*
44 | .....*
45 | ........*
46 | ...........*
47 | .............*
48 | ................*
              EINGABE                POTENZSPEKTRUM
```

Abbildung 5.9 Einfache Grafik in Abacus

```
[A32] "min=
[B32] min(sp)      {Zeilen 9 bis 24}
[A33] sp=wiederh(".",(funk.max-$funk.min)
        * 18/($funk.max-$funk.min+1))+"*"      {Zeilen 33 bis 48}
```

Die zweite Datenserie zeichnet das Leistungsspektrum.

```
[D31] "=max
[C31] max(E28:T28)
[D32] "=min
[C32] 0
[C33] sp=wiederh(".",(index(zle()-28,28)- $pot.min)
        *18/($pot.max-$pot.min+1))+"*"      {Zeilen 33 bis 40}
```

Die dritte Serie von Eingaben bewirkt eine grafische Darstellung der Kosinus-Komponenten.

```
[F31] "= max
[E31] max(E26:T26)
[F32] "= min
[E32] min(E26:T26)
[E33] sp=wiederh(".",(index(zle()-28,26)- $kos.min)
        *18/($kos.max-$kos.min+1))+"*"      {Zeilen 33 bis 40}
```

Die letzte Serie erzeugt die grafische Ausgabe der Sinus-Komponenten.

```
[Y31] "= max
[X31] max(X26:AM26)
[Y32] "= min
[X32] min(X26:AM26)
[X33] sp=wiederh(".",(index(zle()-9,26)- $sin.min)
        *18/($sin.max-$sin.min+1))+"*"      {Zeilen 33 bis 40}
```

Die Transformation benutzen

Wie bereits erwähnt, sollten die Eingabewerte in die Felder B9 bis B24 eingetragen werden. Sie können beliebige Werte wählen, doch hier einige Vorschläge:

```
[B9] sp=10*kos(pi()*(zle()-9)/8)      {Zeilen 9 bis 24}
[B9] sp=10*kos(pi()*(zle()-9)/4)      {Zeilen 9 bis 24}
[B9] sp=10*sin(pi()*(zle()-9)/8)      {Zeilen 9 bis 24}
[B9] sp=10*vorz(kos(pi()*(zle()- 9)/8))      {Zeilen 9 bis 24}
[B9] sp=10      {Zeilen 9 bis 24}
```

Denken Sie daran, daß der Kalkulationsmodus ausgeschaltet ist, und Sie daher die Berechnung der einzelnen Resultate mit **neukalk** anfordern müssen.

Für die Verwendung von Namen als Feldadressen spricht auch die Tatsache, daß man das Fenster leicht auf die interessanten Stellen in der Tabelle verschieben kann, indem man mit **F5** (Nach–Befehl) und einer Namenadresse den Cursor dorthin bringt.

KAPITEL 6

QL ABACUS ÜBERSICHT

FUNKTIONSTASTEN

Zusätzlich zu den allgemeingültig belegten Funktionstasten **F1**, **F2** und **F3** verwendet QL Abacus die beiden Tasten **F4** und **F5** wie folgt:

F4	Mit dem Cursor zwischen zwei Fensterhälften hin– und herwechseln
F5	Mit dem Cursor auf ein bestimmtes Feld springen. (Der Feldname darf nicht mit einem Umlaut oder ß beginnen.)

FELDVERWEISE

Feldverweise auf Einzelfelder, Zeilen, Spalten oder Bereiche können in Form der vorgegebenen Buchstaben– und Zahlenkoordinaten erfolgen oder anhand von benutzerdefinierten Namen.

Einzelfelder

Ein Verweis auf ein Einzelfeld besteht aus zwei Teilen: einer Spalten– und einer Zeilenadresse, wobei die Reihenfolge unwichtig ist, solange keine Verwechslungsgefahr besteht.

Das Abacus-Arbeitsblatt besteht aus 64 Spalten, bezeichnet von A bis BL, und 255 Zeilen, von 1 bis 255. Typische Feldverweise (Feldadressen) sind:

```
A1   AC13   BD200
```

Bereichsverweise

Ein Bereichsverweis setzt sich zusammen aus zwei Feldverweisen, die durch Doppelpunkt getrennt sind. Die erste Adresse bestimmt das Feld in der oberen linken Ecke und die zweite das Feld in der unteren rechten Ecke des Bereichs. Beispiele für Bereichsverweise:

```
B5:D9
AZ23:BA155
```

Zeilen– und Spaltenverweise

Mehrere Felder entlang einer Zeile oder in einer Spalte können als Bereiche aufgefaßt werden, deren eine Dimension nur eine Spalte bzw. eine Zeile umfaßt. Folglich sind Bereichsverweise dieser Art möglich:

`A3:L3` {Felder A bis L in Zeile 3}
`D7:D11` {Felder 7 bis 11 in Spalte D}

Bereichsbezeichner

Es gibt zwei Bereichsbezeichner: **zle()** und **sp()**. Sie beziehen sich auf die Felder der aktuellen Zeile bzw. Spalte (d.h. der Zeile oder Spalte, in deren Schnittpunkt das Feld mit der Formel liegt).

Bei Verwendung der Bereichsbezeichner in einer Formel muß die exakte Dimension, d .h. das Anfangs– und das Endfeld, angegeben werden. Abacus schlägt jeweils 2 Felder vor, die akzeptiert oder überschrieben werden können.

Bereichsbezeichner werden auf zwei Arten eingesetzt. Einmal dienen sie zum Füllen der aktuellen Zeile bzw. Spalte mit:

zle = *(Formel)* bzw. **sp** = *(Formel)*

oder sie bilden das Argument für solche Funktionen, die auf Bereiche angewendet werden, wie etwa **anzahl(zle)**. Selbstverständlich bezeichnen sie immer nur die Felder einer einzigen Zeile oder Spalte.

Die beiden Verwendungsmethoden sind beliebig kombinierbar, z .B.

sp = durchschn*(zle)*

Wann immer sie als Bestandteil einer Formel auftreten, fragt Abacus nach den exakten Bereichsgrenzen.

Relative und absolute Feldadressen

Feldadressen in Abacus sind normalerweise relativ. Das heißt, die Relation, die örtliche Beziehung zwischen dem Feld, welches den Verweis enthält, und dem Feld, auf das verwiesen wird, ist ausschlaggebend. Beim Kopieren eines derartigen Verweises in ein anderes Feld wird dieser so verändert, daß die relative Position der beiden neuen Felder identisch ist mit der ursprünglichen. Das klingt komplizierter als es ist. Stellen Sie sich vor, eine Formel in Feld B2 enthalte einen Verweis auf das Feld A1, welches eine Zeile höher und eine Spalte weiter links liegt. Beim Kopieren der Formel aus Feld B2 auf D4 ändert sich das Bezugsfeld auf C3, weil C3 im Verhältnis zu D4 dieselbe Position einnimmt wie A1 für B2 (nämlich eine Zeile höher, eine Spalte weiter links).

Eine grafische Darstellung dieses Zusammenhangs gibt die Abbildung 6.1. Die Formel in Feld X enthält einen Verweis auf das schwach schattierte Feld. Die Kopie der Formel in Feld Y bezieht sich dann entsprechend auf das dunkel eingefärbte Feld. Die relative Position der beiden Felder in jedem Paar ist die gleiche.

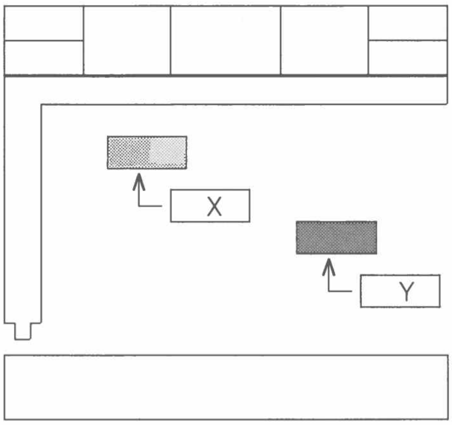


Abbildung 6.1 Relative Feldverweise

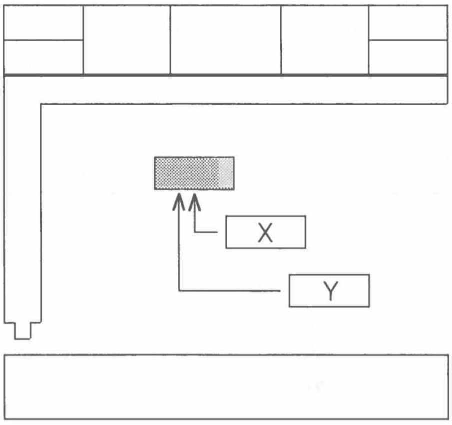


Abbildung 6.2 Absolute Feldverweise

Nehmen wir an, das Feld A2 enthalte die Formel A1 *2, die wir mit Hilfe von **echo** in alle Felder des Bereichs B2:G2 kopieren. Eine Überprüfung des Inhalts der einzelnen Felder ergibt:

Feld	A2	B2	C2	D2	E2	F2	G2
Inhalt	A1 *2	B1 *2	C1 *2	D1 *2	E1 *2	F1 *2	G1 *2

Jeder Feldverweis kann durch ein vorangestelltes Dollarzeichen absolut erklärt werden. Ein so gekennzeichneter Verweis bleibt unverändert, unabhängig davon, wohin er kopiert wird. Beliebige Kopien einer Formel mit dem Verweis $A1 enthalten immer diesen Verweis. Genausogut können auch Feldnamen für absolute Verweise eingesetzt werden, z .B. $Feb.Gew.

Abbildung 6.2 illustriert die Wirkungsweise eines absoluten Feldverweises. Die Formel in Feld X enthält einen absoluten Verweis auf das schattierte Feld. Die Kopie dieser Formel in Feld Y bezieht sich auf dasselbe Feld.

Nehmen wir nochmals das vorherige Beispiel, diesmal jedoch mit einer absoluten Feldadresse. Geben Sie die Formel $A1 *2 in Feld A2 und machen Sie mit **echo** Kopien in die Felder B2 bis G2. Der Inhalt aller Felder ist jetzt derselbe, nämlich:

Feld	A2	B2	C2	D2	E2	F2	G2
Inhalt	$A1 *2	$A1 *2	$A1 *2	$A1 *2	$A1 *2	$A1 *2	$A1 *2

Vergleichen Sie dazu auch die Funktion **index()**.

NAMEN

Zeilen- und Spaltennamen

Ein Name ist ein Feld, das einen alphanumerischen Inhalt aufweist, das heißt Text, bestehend aus Buchstaben und Ziffern. Jedes beliebige Feld dieser Art kann dazu dienen, eine Zeile oder eine Spalte in der Tabelle zu identifizieren (zu benennen). Es ist auch möglich, Namen zur Benennung von Einzelfeldern zu verwenden; hingegen können sie niemals Bereichsverweise ersetzen oder sich auf einen ganzen rechteckigen Block von Feldern beziehen.

Beim Auftreten eines Namens innerhalb eines Ausdrucks oder einer Formel ermittelt Abacus anhand fester Regeln, ob es sich um einen Verweis auf eine Zeile, eine Spalte oder ein Einzelfeld handelt. Die Regeln für Zeilen und Spalten sind:

1 Die Zeile und die Spalte, in deren Schnittpunkt sich der Name befindet, werden nach rechts und nach unten auf einen numerischen Feldinhalt abgesucht.
   a) Wenn nur nach rechts Einträge gefunden werden, bezieht sich der Name auf die Zeile, beginnend bei dem ersten Eintrag.
   b) Wenn nur nach unten Einträge gefunden werden, bezieht sich der Name auf die Spalte, beginnend beim ersten gefundenen Eintrag.
   c) Wenn in beiden Richtungen Einträge gefunden werden, gibt der am nächsten gelegene den Ausschlag (Anzahl dazwischenliegender Felder).

2 Wenn die Regeln unter (1) keine eindeutige Zuordnung gestatten, der fragliche Name jedoch auf der linken Seite eines Ausdrucks vorkommt, dann übernimmt er den Typ des Namens (der Namen), der (die) auf der rechten Seite des Ausdrucks steht (stehen). Wenn z .B. "Kosten" ein Spaltenname ist, und ein Ausdruck

```
Verkauf = Kosten * 0.5
```

lautet, dann gilt "Verkauf" ebenfalls als Spaltenname.

Wenn keine der beiden Regeln zum Ziel führt, reagiert Abacus mit einer Fehlermeldung.

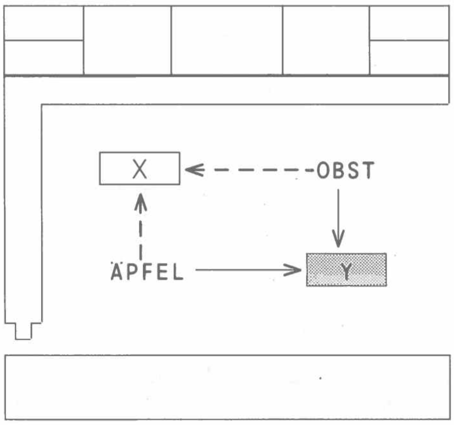


Abbildung 6.3 Benennen eines Feldes

Feldnamen

Zur Identifikation eines Einzelfeldes brauchen Sie zwei Namen. Diese beiden Namen, getrennt durch einen Punkt, bilden den Feldverweis. Bei Vorliegen der beiden Namen OBST und ÄPFEL kann der Feldverweis

```
obst.äpfel
```

lauten oder auf eine eindeutige Abkürzung geschrumpft werden, z .B. "ob.äpf" oder ähnliches.

Ein derartiger Feldverweis bezieht sich auf ein Feld im Schnittpunkt der Spalte und Zeile, welche diese beiden Namen tragen. Aus Abbildung 6.3 geht jedoch hervor, daß es in der Tabelle nicht nur ein Feld gibt, auf das diese Bedingung zutrifft, sondern zwei (X und Y).

Das mit dem Verweis angesprochene Feld ist jenes, welches am weitesten rechts und am weitesten nach unten im Arbeitsblatt liegt. In unserem Beispiel also das Feld Y. Merken Sie sich also, Namen immer oberhalb oder links von den Feldern zu positionieren, auf die sie sich beziehen.

Wenn Sie mit **F5** auf ein bestimmtes Feld springen wollen, dann darf der Feldname nicht mit einem Umlaut oder ß beginnen.

FORMELN

Unter einer Formel versteht man jede zulässige Kombination von Funktionen, Feldverweisen, Namen und Operatoren. Typische Beispiele sind:

```
A1*B3
monat(sp()-1)
wenn(teilkette(B6,"ist"),1,0)
wiederh("=",länge(G23))+":"
```

Stammformeln

Jede neue Formel wird nicht nur auf ein oder mehrere Felder angewendet, sondern außerdem in einer Liste mit Stammformeln gespeichert. Folglich kann jede Stammformel in einem einzigen Feld oder in einer Reihe von Feldern vorkommen. Beim Ausfüllen von Feldern mit Hilfe der Zeilen– und Spaltenbezeichner oder mit **kopie** oder **echo** teilen alle diese Felder eine gemeinsame Stammformel. Etwaige relative Feldverweise werden entsprechend der jeweiligen Feldposition modifiziert, so daß die einzelnen Formeln äußerlich zwar etwas verschieden aussehen, jedoch im Prinzip auf die gleiche Stammformel zurückgehen.

Aus diesem Grund ist es möglich, alle Kopien einer Formel durch bloße Änderung einer einzigen Formel zu ändern – mit Hilfe des **ändern**-Befehls, der gleichzeitig auch die Stammformel und damit alle anderen Formeln, die auf diese zurückgehen, ändert.

DIE BEFEHLE

Es folgen alle verfügbaren Abacus-Befehle in alphabetischer Reihenfolge.

ÄNDERN (Ä)

Mit diesem Befehl kann ein Feld geändert werden. Der Inhalt des Feldes, welches den Cursor enthält, wird in die Eingabezeile kopiert und kann dort mit Hilfe des Zeileneditors modifiziert werden. Der Zeileneditor wird in der Einführung zu den QL-Programmen beschrieben. Durch Drücken von **ENTER** ersetzen Sie die alte Version des Feldinhalts durch die neue.

AUSDRUCK (A)

Dieser Befehl schickt einen spezifizierten Ausschnitt der Kalkulationstabelle entweder an den Drucker oder auf eine Microdrive-Datei. Sie werden zuerst gefragt, ob die in der Tabelle angezeigten Werte oder die zugrundeliegenden Formeln ausgedruckt werden sollen. **ENTER** für angezeigte Werte, **F** für Formeln drücken. Danach folgt die Eingabe über den Bereich, der ausgedruckt werden soll, und darüber, ob der Papierausdruck mit (**ENTER**) oder ohne (**O**) Tabellenrand erfolgen soll. Schließlich ist noch die Frage nach dem Zielort zu beantworten: **ENTER** für Drucker, **D** für Datei. Im letzteren Fall muß noch der gewünschte Dateiname eingegeben werden (gefolgt von **ENTER**).

Darauf wird der spezifizierte Ausschnitt des Arbeitsblatts an den Bestimmungsort geschickt. Der Druckvorgang kann jederzeit mit **ESC** gestoppt werden.

Wurde **F** für Formeln gewählt, druckt Abacus zunächst eine numerierte Liste aller im Arbeitsblatt vorkommenden Formeln und druckt dann das Arbeitsblatt selbst aus, wobei alle Felder, die eine Formel enthalten, mit der entsprechenden Nummer versehen sind.

Wenn Sie bei der Angabe des Microdrive-Dateinamens als Bestimmungsort keinen Zusatz (keinen "Nachnamen") angeben, nimmt Abacus **_lis** an.

BÜNDIG (B)

Dieser Befehl dient zur Positionierung von Text und Zahlen innerhalb der Felder eines Bereichs. Er verfügt über die beiden Hauptoptionen "besetzte Felder" und "leere Felder" (leere Felder dient zur Änderung der Standardvorgabe, gilt also auch für Felder, die im Moment noch ohne Inhalt sind). **ENTER** wählt "besetzte Felder", **L** "leere Felder".

Anschließend muß zwischen Text (**ENTER**) und Zahlen (**Z**) gewählt werden, worauf bei beiden mit **ENTER** linksbündige, mit **M** mittige (zentrierte) und mit **R** rechtsbündige Positionierung festgelegt werden kann.

Bei der Modifikation "besetzter Felder" wird außerdem nach dem exakten Bereich gefragt, der von der Modifikation betroffen werden soll.

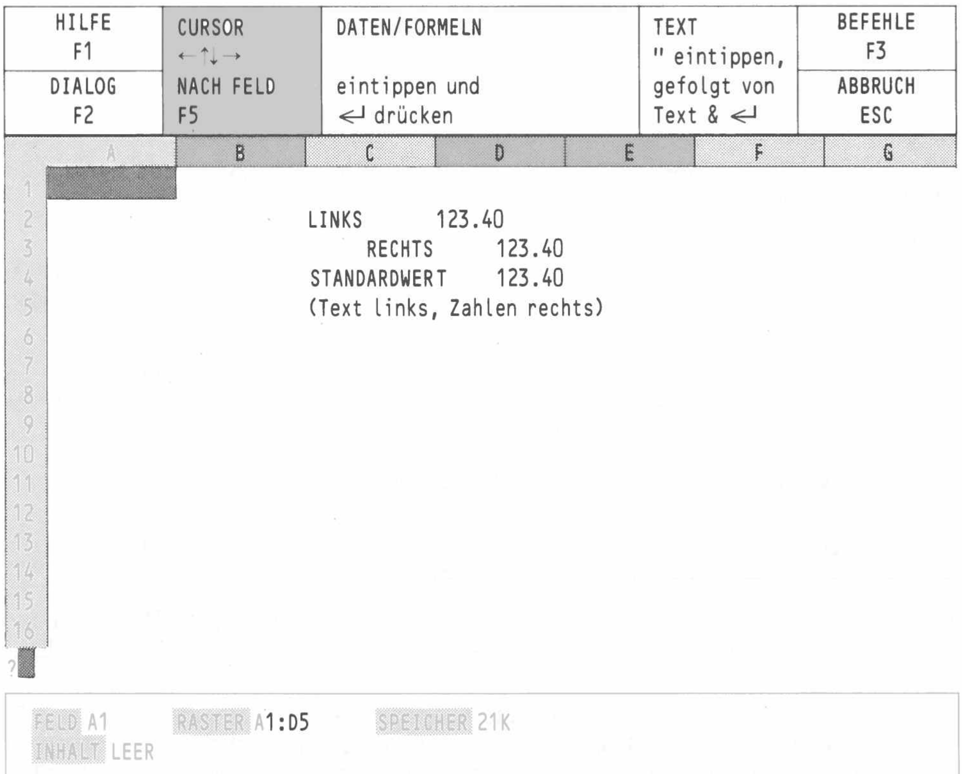


Abbildung 6.4 Rechts– und linksbündige Anordnung

Diese Angabe ist bei der Änderung der Standardvorgabe (leere Felder) überflüssig, da sie automatisch für alle Felder Geltung hat, in die Neueintragungen vorgenommen werden – bis zu dem Zeitpunkt, wo eine erneute Änderung an dieser Einstellung gemacht wird.

Eine Reihe verschiedener Anordnungen (Justierungen), zusammen mit den standardmäßig vorgesehenen Einstellungen – Text linksbündig, Zahlen rechtsbündig – sehen Sie in Abbildung 6.4.

DATEIEN (D)

Dieser Befehl ermöglicht die Modifikation von Abacus-Dateien, die bereits auf Microdrive-Kassette gespeichert sind. Die einzelnen Optionen aus dem Dateien-Menü fragen nach dem Dateinamen. Durch Eingabe von **?** können Sie sich jederzeit einen Katalog aller Dateien auf Microdrive 2 auflisten lassen.

Das Dateien-Menü präsentiert die folgenden Optionen:

Reserve
erstellt eine Zweitkopie einer Abacus–Datei. Sie werden nach dem Namen der Datei gefragt, die kopiert werden soll. Wir empfehlen Ihnen unbedingt, von allen Dateien Reservekopien anzufertigen, um sich vor Datenverlust durch versehentliches Überschreiben oder Beschädigung der Kassette zu schützen.

Löschen
löscht die benannte Datei von einer Microdrive-Kassette. **Vorsicht: Dieser Befehl kann nicht rückgängig gemacht werden und muß daher sorgfältig überlegt werden.**

Export
exportiert die genannte Datei. Sie wird in einem für den Import in die Programme Archive, Easel oder Quill geeigneten Format gespeichert.

Abacus fragt zuerst nach dem Programm, in das exportiert werden soll, Quill, Archive oder Easel. Quill mit **ENTER**, Archive mit **A** und Easel mit **E** anwählen.

Danach muß der Bereichsverweis für den zu exportierenden Tabellenausschnitt eingegeben und mit **ENTER** abgeschlossen werden.

Der Export nach Archive oder Easel kann zeilen– oder spaltenweise erfolgen. Abacus schlägt zeilenweisen Export vor, was Sie mit **ENTER** bestätigen, bzw. mit **S** auf Spalten umstellen. Beim Export nach Quill bleibt Ihnen keine andere Wahl: hier wird immer zeilenweise exportiert.

Schließlich bittet Abacus noch um einen Namen für die Exportdatei. Bei Fehlen eines Zusatzes (eines Nachnamens) wird **_exp** angehängt.

Formatieren
formatiert die Kassette im Microdrive 2. Die Microdrive-Bezeichnung mdv2_ wird automatisch vorgegeben; anschließend ist der Name des Datenträgers zu spezifizieren.

Vorsicht: Vergewissern Sie sich, daß die Kassette im Microdrive 2 keine Dateien enthält, die Sie vielleicht noch brauchen. Beim Formatieren gehen alle Dateien unwiederbringlich verloren.

Import
importiert eine benannte Datei. Erlaubt Abacus das Einlesen von Dateien, die aus Archive oder Easel stammen. Für eine ausführliche Beschreibung der Import-Funktion verweisen wir auf den Informationsteil.

Dateien können zeilen– oder spaltenweise importiert werden, ganz nach Bedarf. Es muß die Adresse des Feldes in der oberen linken Ecke des Tabellenausschnitts angegeben werden, in den die Daten importiert werden sollen.

Bei fehlendem Zusatz (Nachname) im Dateinamen nimmt Abacus _exp an.

ECHO (E)

Dieser Befehl erstellt eine Kopie der Daten oder der Formel in einem bestimmten Feld und setzt sie in alle Felder des genannten Bereichs.

Sie geben entweder ausdrücklich eine Feldadresse an oder drücken **ENTER**, um das aktuelle Feld zu kopieren. Danach folgt die Angabe des Bereichs, gefolgt von **ENTER**.

FORMAT (F)

Mit den Optionen dieses Befehls wird das Aussehen des Arbeitsblatts bestimmt, z.B. das Anzeigeformat auf 80, 64 oder 40 Spalten festgelegt, je nachdem, ob mit einem Monitor oder einem Fernseher gearbeitet wird. Die einmal eingestellten Optionen gelten bis zu einer erneuten Änderung bzw. bis zum Verlassen von Abacus. Beim Abspeichern

(Sichern) einer Kalkulationstabelle werden alle gewählten Optionen, mit Ausnahme des Anzeigeformats, mit abgespeichert und beim erneuten Laden desselben Arbeitsblatts wieder eingelesen.

Die vorgenommenen Veränderungen beziehen sich jedoch immer nur auf eine bestimmte Anwendung, d.h. auf ein Arbeitsblatt, niemals auf Abacus als Programm. Sie sind also gegebenenfalls bei jedem Umsteigen von SuperBASIC auf Abacus neu einzurichten.

Auto-Kalkulation bei Eingabe
zum Ein- und Ausschalten der automatischen Neukalkulation. Die Auswahlzeile funktioniert wie ein Kippschalter zwischen Ja und Nein.

Im Auto-Kalkulationsmodus (Ja) wird das ganze Arbeitsblatt bei jeder Eingabe ganz neu durchkalkuliert. Im andern Fall (Nein) erfolgt eine Kalkulation erst bei Ausführung des **neukalk**-Befehls. Standardvorgabe ist Ja.

Leer wenn Null
wechselt zwischen zwei Methoden der Behandlung von Nullwerten in der Tabelle. Die Standardvorgabe ist Anzeige von Null im normalen Anzeigeformat des betreffenden Feldes; die Alternative ist Leerlassen des Feldes, wenn der Inhalt Null wird.

Beachten Sie, daß bei Setzen der alternativen Einstellung das Feld nur dann als leeres Feld angezeigt wird, wenn es einen echten Nullwert enthält, nicht jedoch, wenn der Wert z .B. bei einem zweistelligen Dezimalformat 0.003 ist. In diesem Fall würde 0.00 angezeigt.

Berechnungsreihenfolge
wählt zwischen Berechnung der Tabelle nach Zeilen (ZLE) und nach Spalten (SP). Drücken von **B** wechselt zwischen diesen beiden Methoden. Die eingestellte Option gilt sowohl für Auto-Kalkulation wie auch für Neukalkulation mit **neukalk**. Standardvorgabe ist zeilenweise Berechnung, welche erheblich schneller arbeitet als die spaltenweise.

80,64,40 Spalten-Anzeige
wählt die Anzahl Spalten (Zeichen) je Bildschirmzeile. Eingabe von **8, 6** oder **4** (gefolgt von **ENTER**), bestimmt 80, 64 oder 40er Format. Der Anfangswert ist entweder 80 oder 64, je nachdem, ob Sie beim Laden von Abacus die Monitor- oder die TV-Option gewählt haben.

Formular-Vorschub zwischen Seiten
bestimmt, ob jeweils nach einer Seite Druckausgabe ein Formularvorschub eingelegt wird oder nicht. Schaltet zwischen Ja und Nein hin und her; Standardvorgabe ist Ja.

Zeilenabstand beim Ausdruck
setzt den Abstand zwischen den Textzeilen für den Papierausdruck. Die Zahlen beziehen sich auf die Anzahl Leerzeilen. 0, 1 oder 2 eingeben, **ENTER** erübrigt sich. Doppelzeilige Druckausgabe wird durch Eingabe von **1** (= 1 Leerzeile zwischen zwei Textzeilen) erzielt. Standardvorgabe ist 0.

Seitenlänge Druckerpapier (Zeilen)
spezifiziert die Anzahl Zeilen pro Druckseite. Eine Zahl eingeben und **ENTER** drücken. Standardvorgabe ist 66, maximale Einstellung ist 255.

Währung
bestimmt das Währungssymbol für Geldbeträge in der Tabelle. **ENTER** nicht notwendig. Akzeptiert nur Einzelzeichen. Standardvorgabe ist das Dollarzeichen.

Druckerpapier-Breite (Zeichen)
bestimmt die Druckbreite, d.h. gedruckte Zeichen je Zeile bei der Druckausgabe. Eine Zahl und **ENTER** eingeben. Standardvorgabe ist 80, maximale Einstellung 255.

KOPIE (K)

Dieser Befehl dient dazu, einen Block von Feldern an eine andere Stelle in der Tabelle zu kopieren. Abacus fragt nach dem Bereich, der kopiert werden soll (Feld obere Ecke links:Feld untere Ecke rechts, z .B. A1:B3). Eingabe mit **ENTER** abschließen. Danach das linke obere Feld des neuen Bereichs angeben, an den die Kopie übertragen werden soll. Der Kopiervorgang wird bei Drücken von **ENTER** ausgeführt.

LADEN (L)

Befehl zum Laden einer Datei von Microdrive-Kassette. Zuerst wird der gewünschte Dateiname eingegeben. Durch Eingabe von **?** holen Sie sich eine Liste aller Dateien in Microdrive 2 auf den Bildschirm.

Wenn Sie im Namen keinen Zusatz spezifizieren, nimmt Abacus den Nachnamen **_aba** an.

MISCHEN (M)

Dieser Befehl dient zur Kombination oder Konsolidierung von Daten aus einer bereits früher auf Microdrive-Kassette gesicherten Datei mit dem laufenden Arbeitsblatt. Als erstes werden Sie um den Dateinamen gebeten, der eingelesen werden soll, und dann müssen Sie angeben, ob diese neuen Daten zu den Daten des aktuellen Arbeitsblattes addiert (**ENTER**) oder von diesen subtrahiert (**S**) werden sollen.

Immer wenn ein Feld aus der neu eingelesenen Datei mit einem Feld in der aktuellen Tabelle gepaart werden kann, wird der neue Wert zum vorhandenen addiert bzw. davon subtrahiert. Andere Feldinhalte bleiben unverändert. Der Befehl hat keine Wirkung auf Namen-Felder; diese sind folglich gegen jede Änderung geschützt.

Das resultierende Arbeitsblatt enthält in den betroffenen Feldern nur numerische Werte; die diesen zugrundeliegenden Formeln sind vernichtet, da sie in der konsolidierten Tabelle keinen Sinn ergeben würden.

Der **mischen**-Befehl bietet eine schnelle und einfache Methode, die Daten aus zwei gleich strukturierten Tabellen miteinander zu verschmelzen. Dabei muß allerdings sehr genau darauf geachtet werden, daß die einzelnen Feldeingaben genau miteinander übereinstimmen, weil sonst kein befriedigendes Resultat erwartet werden kann.

NEUKALK (N)

Dieser Befehl erzwingt eine Neukalkulation aller Formeln in der aktuellen Tabelle. Normalerweise wird bei jeder einzelnen Eingabe eine vollständige Kalkulation durchgeführt, so daß dieser Befehl nur zur Anwendung kommt, wenn der Auto-Kalkulationsmodus ausgeschaltet wurde, bzw. um die Funktionen **anfn()** oder **anft()** zu aktivieren.

ÖFFNUNG (Ö)

Mit diesem Befehl kann gewählt werden zwischen einem einzelnen Fenster und zwei Fenstern zur Anzeige von verschiedenen Ausschnitten der Tabelle.

Als erstes entscheiden Sie zwischen vertikaler und horizontaler Öffnung bzw. Rückkehr auf ein einzelnes Fenster. Um von einer horizontalen zu einer vertikalen Teilung umzuschalten, müssen die beiden Fenster erst zusammengelegt und dann erneut eine Trennung vorgenommen werden.

Die Teilung erfolgt immer an der Cursorposition. Plazieren Sie also zuerst den Cursor entsprechend. Es werden immer ganze Spalten angezeigt, und die minimale Breite jedes Fensters bei einer vertikalen Teilung ist 10 Zeichen.

Ferner können die beiden Fenster wahlweise entweder zusammen oder separat bewegt werden. Bei Option **Z** bewirkt jede Änderung in der Position des einen Fensters – parallel zur Teilung – eine entsprechende Änderung in der Position des anderen. Bewegungen im rechten Winkel zu der Teilungslinie sind nicht in dieser Weise miteinander verknüpft. Die Option **S** ermöglicht es, beide Fenster unabhängig voneinander im Arbeitsblatt umherzuschieben.

ORDNEN (O)

Dieser Befehl dient zum Ordnen der Tabellenzeilen in aufsteigender Folge, wobei der Sortierschlüssel der Inhalt einer spezifizierten Spalte ist.

Als erstes geben Sie die Spalte ein, nach der sortiert werden soll, dann die erste und die letzte zu sortierende Zeile. Die genaue Reihenfolge ist:

Leere Felder
Numerische Felder, in absteigender numerischer Folge
Textfelder in alphabetischer Ordnung

Verwenden Sie **ordnen** nur für Felder, die Daten enthalten. Auf Formeln kann sich der Vorgang ungünstig auswirken, weil keine Anpassungen an die örtliche Verschiebung vorgenommen werden.

RASTER (R)

Dieser Befehl führt Änderungen durch, welche die ganze Kalkulationstabelle betreffen: Einfügen und Löschen von Spalten und Zeilen und Modifikation der Spaltenbreite.

Die drei Optionen im raster-Menü sind:

Einfügen
schiebt leere Zeilen oder Spalten in die Tabelle ein. Die erste Anfrage betrifft Zeilen (**ENTER**) oder Spalten (**S**), die zweite die Position und die dritte die Anzahl der einzufügenden Zeilen bzw. Spalten. Nach dem abschließenden **ENTER** wird die Einfügung direkt vor der spezifizierten Position vorgenommen. Dafür gehen am Ende der Tabelle die entsprechende Anzahl Spalten oder Zeilen verloren, falls diese mit Daten besetzt sind.

Das heißt, wenn Sie drei Spalten oder Zeilen irgendwo in der Tabelle einfügen, gehen ebenfalls drei Spalten oder Zeilen am Schluß der Tabelle verloren – mit allen darin befindlichen Daten. Sie können nicht wiederhergestellt werden (außer durch nochmalige Eingabe).

Löschen
dient zum Löschen von einer oder mehreren Zeilen oder Spalten aus der Tabelle. Auf die erste Anfrage werden Zeilen (mit **ENTER**) spezifiziert oder Spalten (mit **S**), dann folgt die Anfrage VON mit einem Anfangspunkt und BIS mit dem Endpunkt des zu löschenden Bereichs, gefolgt von **ENTER**.

Mit dem abschließenden **ENTER** führen Sie den Befehl aus, wobei die betreffenden Zeilen oder Spalten gelöscht werden und die folgenden nachrücken, um die Lücke zu schließen. Entsprechend werden am rechten bzw. am unteren Ende der Tabelle neue leere Zeilen oder Spalten eingefügt, so daß die Tabelle ihre normale Dimension beibehält.

In beiden Optionen, **einfügen** und **löschen**, werden alle Formeln entsprechend ihrer neuen Position modifiziert.

Spaltenweite
ändert die Anzahl Zeichenpositionen in einer oder mehreren Spalten. Geben Sie zuerst die Anzahl Zeichen ein und dann den Anfangs- und Endpunkt der Änderung.

SICHERN (S)

Befehl zum Sichern (Speichern) einer Datei auf Microdrive-Kassette. Zunächst werden Sie nach dem Dateinamen gefragt. Drücken von **?** bewirkt eine Auflistung aller Dateien auf Microdrive 2.

Wenn Sie dem Dateinamen keinen Zusatz anhängen, nimmt Abacus den Nachnamen **_aba** an.

TILGEN (T)

Dieser Befehl leert alle Feldinhalte aus dem Arbeitsblatt und bringt Sie in den Anfangszustand von Abacus (Neustart) zurück. Da es sich hier um einen drastischen Befehl handelt, dessen Wirkung nicht mehr rückgängig zu machen ist, wenn er einmal ausgeführt wurde, fordert Abacus sicherheitshalber eine Bestätigung. Sie haben Gelegenheit, den Befehl mit **ESC** noch zu annullieren und wieder ins Befehlsmenü zurückzukehren. **ENTER** bewirkt Ausführung des Tilgungsbefehls.

VERLASSEN (V)

Dieser Befehl dient zum Verlassen von Abacus nach Beendigung einer Arbeit.

Bei diesem Vorgang geht die aktuelle Tabelle verloren. Sicherheitshalber müssen Sie diese Entscheidung noch einmal bestätigen, bzw. können es sich anders überlegen und mit **ESC** in das Arbeitsblatt zurückkehren. **ENTER** bewirkt den Ausstieg aus Abacus und Rückkehr zu SuperBASIC.

WEGNEHMEN (W)

Dieser Befehl dient zum Radieren oder Löschen des Inhalts eines oder mehrerer Tabellenfelder. Es wird ein Bereichsverweis angefordert. Alle Felder in dem spezifizierten Bereich werden gelöscht.

ZAHLEN (Z)

Dieser Befehl dient zur Änderung des Anzeigeformats eines einzelnen Feldes oder eines Bereichs, ohne die Werte in irgendeiner Weise zu beeinflussen.

Die erste Option betrifft den Wirkungsbereich des Befehls: Änderung der besetzten Felder (**ENTER**) oder Änderung der leeren Felder (d.h. der Standardvorgaben) mit **L**.

In beiden Fällen werden dann die verschiedenen Anzeigeformate präsentiert:

Dezimal
Zahlen werden in einer Festkommanotation dargestellt, d.h. alle Werte weisen dieselbe Anzahl Dezimalstellen auf (Stellen nach dem Komma bzw. nach dem Punkt). Zahlen mit mehr Dezimalstellen werden auf- oder abgerundet. Sie werden nach der Anzahl Dezimalstellen gefragt; Abacus akzeptiert nicht mehr als 14.

Wenn Sie auch die Tabellenwerte (Inhalte) runden wollen (also nicht nur die Anzeige im Feld), müssen Sie die Aufrundung selbst durchführen. Hier das Vorgehen zum Runden auf zwei Dezimalstellen:

1. mal 100 rechnen (mal 1000 für 3 Dezimalstellen, usw.)
2. 0.5 addieren
3. Dezimalbruch mit der **ganzzahl()**-Funktion abschneiden
4. durch 100 (bzw. 1000) teilen.

Die folgende Formel rundet den Wert in Feld C3 auf 2 Dezimalstellen:

```
ganzzahl(C3*100+0.5)/100
```

Ganzzahl
Mit dieser Option werden alle Zahlen als ganzzahlige Werte dargestellt – wie mit der Funktion **ganzzahl()**. Negative Werte können auf Wunsch als Minuszeichen (**ENTER**) oder in Klammer (**K**) angezeigt werden.

Wenn nicht nur die Anzeige, sondern auch der eigentliche Wert (Inhalt) der Felder ganzzahlig gemacht werden soll, ist die Funktion **ganzzahl()** zu verwenden.

Exponent
Diese Option zeigt Zahlen im Exponentialformat an, eine Notation, die in der Wissenschaft häufig verwendet wird, wenn es um sehr kleine und sehr große Zahlen geht. Sie werden nach der Anzahl der Dezimalstellen gefragt, wobei 14 das Maximum ist. Auch hier wird gegebenenfalls eine Rundung vorgenommen.

Prozent
zeigt Zahlen als Prozentsätze an, z.B. 0.55 als 55%. Auch hier werden Sie nach der Anzahl der Dezimalstellen gefragt. Maximal 14 sind zulässig.

Allgemein
Diese Option spezifiziert das allgemeine numerische Format. Hier versucht Abacus, die optimale Darstellungsanzeige für jeden Wert im Hinblick auf den verfügbaren Platz zu finden und macht dabei Gebrauch von allen bisher beschriebenen Formaten.

Währung
Bei dieser Option werden die Zahlen im Festkomma-Dezimalformat angezeigt, mit zwei Dezimalstellen und einem vorangehenden Währungssymbol. Negative Werte können auf Wunsch in Klammer gesetzt werden (**K**); **ENTER** bewirkt Anzeige mit Minuszeichen.

Wenn die Option "besetzte Felder" gewählt wird, stellt Abacus bei allen vorangehend beschriebenen Formaten die Frage nach dem Geltungsbereich der Änderung. Ihre Antwort kann jede Art von Feld- oder Bereichsverweis enthalten (auch Namen oder Bereichsbezeichner). Der Verweis ist mit **ENTER** abzuschließen.

Im Gegensatz dazu wird bei Änderung der Standardoptionen (leere Felder) nicht nach einem Geltungsbereich gefragt, da hier die neuen Anzeigeformate automatisch für jede neue Eingabe Anwendung finden.

FUNKTIONEN

Stellen Sie sich eine Funktion am besten als eine Art Rezept vor, das mit einem oder mehreren Werten, den sog. *Argumenten*, bestimmte Operationen ausführt und ein Ergebnis liefert, den *Funktionswert*. In Abacus wird dieser Wert im Tabellenfeld angezeigt, dessen Inhalt eine Funktion ist.

Abacus verfügt über Funktionen, die zwei oder drei Argumente, oder auch nur eines oder gar keines, übernehmen können. Diese werden jeweils direkt nach der Funktion in Klammer angegeben. Zwischen dem Funktionsnamen und der Eröffnungsklammer darf kein Leerzeichen gelassen werden; hingegen sind Leerzeichen zwischen den Werten in der Klammer gestattet. Mehrere Argumente sind durch Kommas voneinander zu trennen. Allen Funktionen muß ein Klammerpaar folgen, auch wenn sie keine Argumente enthalten. Dies hilft, eine Funktion als solche zu identifizieren.

Bei den nachfolgenden Beschreibungen der Abacus-Funktionen steht

n	für einen numerischen Ausdruck oder einen Verweis auf ein Feld, welches einen numerischen Wert anzeigt
text	für einen Textausdruck oder einen Verweis auf ein Feld, welches einen Textwert enthält
bereich	für einen Feldbereich in der Tabelle

Ein *numerischer Ausdruck* ist entweder ein Wert oder ein Ausdruck, der ein numerisches Resultat ergibt.

Ein *Textausdruck* ist entweder eine Textkette (alphanumerische Zeichen, eingeschlossen in Anführungszeichen) oder ein Ausdruck, dessen Resultat eine alphanumerische Zeichenkette ist.

Es folgt eine alphabetische Zusammenstellung aller in Abacus verfügbaren Funktionen.

ABS(*n*) liefert den absoluten Wert ihres Arguments (d.h. den Wert ohne Berücksichtigung eines etwaigen Minuszeichens)

Beispielsweise ergibt **abs(3)** 3 und **abs(– 7)** 7.

ANFN(*Text*) Funktion zur Eingabe numerischer Daten. Sie gibt den *Text* (maximale Länge 40 Zeichen) als Aufforderung zur Benutzereingabe am Bildschirm aus, gefolgt von einem Fragezeichen, und wartet auf eine Antwort über die Tastatur. Diese Eingabe wird dann im Feld, das die Funktion enthält, angezeigt. Die Aufforderung erfolgt bei der ursprünglichen Eintragung der Formel und danach bei Kalkulation der Tabelle mit **neukalk**, nicht jedoch während der normalen automatischen Neukalkulation.

ANFT(*Text*) Funktion zur Eingabe von Text (Zeichenketten). Analog zu **anfn()**, nur daß statt numerischer Eingabe Text erwartet wird.

ANZAHL(*Bereich*)
Gibt die Anzahl der gefüllten Felder im spezifizierten Bereich aus. Gezählt werden nur numerische Felder.

BOG(*n*) Wandelt einen in Grad ausgedrückten Winkel in Bogenmaß um.

BREITE() Gibt die Spaltenbreite des aktuellen Feldes in Anzahl Zeichenpositionen an. Beachten Sie, daß zwei Felder immer durch ein Leerzeichen getrennt sind.

CODE(*Text*) Liefert den ASCII-Code des ersten Zeichens im angegebenen Textargument.

DATUM(*n*) Gibt das aktuelle Datum als Zeichenkette aus, wobei drei Formen möglich sind:

n	Datumszeichenkette
0	"JJJJ/MM/TT"
1	"TT/MM/JJJJ"
2	"MM/TT/JJJJ"

Voraussetzung ist das Setzen der internen System-Uhr (s. Befehle zu SuperBASIC).

DURCHSCHN(*Bereich*)
Gibt als Resultat den Durchschnittswert aller numerischen Werte aus dem angegebenen Bereich aus. Leer- und Textfelder werden bei der Berechnung ignoriert. Sind im ganzen Bereich keine numerischen Werte enthalten, gibt die Funktion den Wert Null aus.

EXP(*n*) Liefert den Wert von e (ca. 2,718) hoch n, wobei das Resultat verfälscht ist, wenn n außerhalb des Bereichs –87 bis +88 liegt, weil dies den numerischen Bereich von Abacus übersteigt.

GANZZAHL(*n*) Liefert den ganzzahligen Teil ihres Arguments und schneidet etwaige Dezimalstellen ab. **Die Rundung erfolgt immer in Richtung 0,** z.B.:

`ganzzahl(3.7)` liefert 3
`ganzzahl(-4.8)` liefert –4

GNW(*Bereich,Prozent,Zeitraum*)
Prozent= n
Zeitraum= n

Berechnet den gegenwärtigen Nettowert einer zukünftigen regelmäßigen Zahlung im angegebenen Bereich. *Prozent* ist der Jahreszinssatz (14 bedeutet 14%). Es wird davon ausgegangen, daß die Daten in *Bereich* sich auf jeweils gleichlange Zeiträume beziehen.

Der gegenwärtige Nettowert ist der aktuelle Geldbedarf zur Sicherstellung einer gegebenen künftigen regelmäßigen Zahlung, unter Annahme eines Zinssatzes für den gegenwärtigen Nettowert. (Barwert bei Rentenrechnungen). Angenommen, Sie haben die Gelegenheit, für eine einmalige Zahlung von DM 70000 einen Laden auf 10 Jahre zu mieten, der im Moment ein jährliches Nettoeinkommen von 10000 einbringt, welches erfahrungsgemäß im Jahr um 10% ansteigt. Alternativ können Sie Ihre DM 70000 für 14% Zins anlegen. Wie entscheiden Sie sich?

Sie müssen den gegenwärtigen Nettowert des zukünftigen Einkommens berechnen und diesen mit dem Betrag vergleichen, den man von Ihnen als Zahlung haben will.

```
[A1] "Fluß
[A2] 0
[A3] 10000
[A4] sp=A3*1.1            {Zeilen 4 bis 12}
[A14] gnw(fluß,14,12)        {Zeilen 2 bis 12}
```

Abbildung 6.5 zeigt das Ergebnis.

```
   |       A       |       B
 1 | Fluß
 2 |          0.00
 3 |      10000.00
 4 |      11000.00
 5 |      12100.00
 6 |      13310.00
 7 |      14641.00
 8 |      16105.10
 9 |      17715.61
10 |      19487.17
11 |      21435.89
12 |      23579.48
13 |
14 |      75088.51
```

Abbildung 6.5 Gegenwärtiger Nettowert

Der gegenwärtige Nettowert (Barwert) in Feld A14 ist höher als der Betrag, den Sie auslegen müssen – also ein interessanter Vorschlag.

Der erste Posten in der Aufstellung bezieht sich auf den Zeitraum Null, der zweite auf den Zeitraum Eins usw. Dies stimmt überein mit der Annahme der Funktion, daß die Zahlung jeweils am Ende jedes Zeitraums hereinfließt. Folglich müssen Sie sich etwas gedulden, bis Ihre Investition etwas einbringt. In der Praxis würde wohl eine Monatsbasis angenommen.

GRAD*(n)* Zur Umwandlung von Bogenmaß in Grad.

IER*(Bereich,Zeitraum)*

Zeitraum = *n*

Berechnet die interne Ertragsrate für die numerischen Daten im angegebenen Bereich, der eine Spalte oder eine Zeile sein kann.

Die in dem Bereich enthaltenen Daten stellen Zahlungen dar für eine Reihe von Zeiträumen, welche jeweils n Monate auseinanderliegen. Negative Werte sind Barausgaben, positive Bareinnahmen.

Die Funktion berechnet auf der Grundlage des Barwerts und des Endwerts den Zinssatz.

Nehmen wir an, man macht Ihnen den Vorschlag, eine sofortige Barzahlung von DM 100 000 zu leisten und dafür während der nächsten 7 Jahre jeweils am Jahresende DM 20000 einzukassieren. Was halten Sie davon?

```
[A1] "Fluß
[A2] -100000
[A3] sp=20000        {Zeilen 3 bis 9}
```

Den Datenbereich können wir mit "Fluß" bezeichnen; der zeitliche Abstand zwischen den Zahlungen beträgt 12 Monate:

```
[C2] ier(fluß,12)       {Zeilen 2 bis 9}
```

Die vollständige Tabelle sollte der Abbildung 6.6 entsprechen und als Resultat eine Ertragsrate von 9,10% ausgeben. Wenn Ihnen jemand für Ihre DM100000 eine vorteilhaftere Verzinsung bietet, lohnt sich das vorgeschlagene Geschäft nicht.

Beachten Sie, daß der erste Posten in der Aufstellung sich auf den Zeitraum 0 bezieht, der nächste auf 2, usw. Die Funktion geht von der Annahme aus, daß die Rückzahlung jeweils am Jahresende fällig ist.

	A	B	C
1	Fluß		
2	-100000.00	9.10	
3	20000.00		
4	20000.00		
5	20000.00		
6	20000.00		
7	20000.00		
8	20000.00		
9	20000.00		

Abbildung 6.6 Interne Ertragsrate (Verzinsung)

INDEX*(Spalte,Zeile)*
Gibt den Inhalt des Feldes im Schnittpunkt der spezifizierten Spalte und Zeile aus. Auch bei absoluter Adressierung ist für Spalte eine Zahl oder ein numerischer Ausdruck einzusetzen.

KETTE*(n,Typ,Dst)*

num = n
Typ = n
Dst = n

Wandelt eine Zahl n in die entsprechende Zeichenkette um. Unter Typ ist das Format gemeint, nämlich:

0 = Dezimal (Gleitkomma)
1 = Exponential
2 = Ganze Zahl (Integer)
3 = Allgemein

Der dritte Parameter, Dst, spezifiziert die Anzahl der Nachkommastellen in der umgewandelten Kette und muß immer angegeben werden, auch wenn die Formate Integer und Allgemein davon keinen Gebrauch machen.

KOS*(n)* Liefert den Kosinus des gegebenen Winkels (in Bogenmaß).

LÄNGE*(Text)* Gibt die Anzahl Zeichen im spezifizierten Text aus.

LN*(n)* Gibt den natürlichen Logarithmus (zur Basis e) der Zahl n im Argument. Ein Null– oder Negativwert von n provoziert eine Fehlermeldung, da keine Logarithmen für diesen Wertebereich definiert sind.

MAX*(Bereich)* Liefert als Ergebnis den höchsten numerischen Feldinhalt aus dem spezifizierten Bereich. Falls keine numerischen Werte angetroffen werden, ist das Ergebnis Null.

MIN*(Bereich)* Liefert als Ergebnis den kleinsten numerischen Feldinhalt aus dem spezifizierten Bereich. Falls keine numerischen Werte angetroffen werden, ist das Ergebnis Null.

MONAT*(n)* Gibt den Monatsnamen, der n entspricht, als Textwert an.

Beispielsweise liefert **monat(3)** die Zeichenkette "März".

Bei Eingabe eines Arguments, das größer als 12 ist, wird n durch 12 geteilt, und der Rest der Division ist ausschlaggebend: **monat(13)** und **monat(1)** ergeben beide "Januar".

NUMWERT*(Text)*
Wandelt das Textargument in den entsprechenden numerischen Wert um; kann nur Text interpretieren, der aus zulässigen numerischen Zeichen besteht und stoppt mit der Umwandlung, sobald ein Zeichen angetroffen wird, das keine numerische Interpretation zuläßt. Beispiel:

`numwert("2.2ABC")` liefert den Wert 2.2, während `numwert("ABC")` 0.0 ausgibt.

PI() Liefert den Wert der mathematischen Konstanten π

QWURZEL(*n*) Gibt die Quadratwurzel ihres Arguments n aus, welches keine negative Zahl sein darf.

SIN(*n*) Liefert den Sinus des Winkels n (in Bogenmaß).

SP() Liefert die Spaltennummer des aktuellen Feldes.

SUCHEN(*Bereich,Absetzung,n*)

Absetzung (offset) = n
Wert = n

Liefert einen Wert aus einem zweiten Bereich, der parallel zu dem angegebenen *Bereich* liegt, jedoch um *Absetzung* Felder von ihm entfernt ist. Der *Bereich* kann eine Zeile oder eine Spalte sein. Die Funktion findet den größten Wert in *Bereich*, der *n* nicht übersteigt. Sie liefert den entsprechenden Wert aus dem zweiten Bereich. Die Werte in *Bereich* müssen in aufsteigender Folge geordnet sein. Die Funktion arbeitet nur dann korrekt, wenn beide Bereiche nur numerische Werte enthalten.

Beispiel: (Abbildung 6.7) SUCHEN (A2:A6,3,4) liefert den Wert 25.

	A	B	C	D	E
1					
2	1				10
3	2				15
4	3				20
5	4				25
6	5				30
7					
8					

Abbildung 6.7: SUCHEN (A2:A6,3,4) liefert den Wert 25

SUMME(*Bereich*)

Der als Resultat ausgegebene Wert ist die Summe der exakten Inhalte der Felder im spezifizierten Bereich, d.h. etwaige Rundungen aufgrund des **zahlen**-Befehls werden ignoriert. Für zwei Felder mit den Werten 3.44 und 9.73 liefert **summe()** den Wert 13.17. Wenn danach Dezimalformat mit einer Nachkommastelle gewählt wird, lauten die beiden Zahlen 3.4 und 9.7, so daß die Summe, deren Wert nach wie vor 13.17 beträgt, auf den scheinbar fehlerhaften Wert 13.2 gerundet wird. Näheres dazu unter dem Befehl **zahlen**.

TAGE(*Text*) Liefert die seit dem 1. Januar 1583 verflossenen Tage bis zu einem als Textausdruck gegebenen Datum von der Form "JJJJ/MM/TT". Die Umwandlung geht vom Gregorianischen Kalender aus und gilt folglich nur für Daten nach 1582.

TAN(*n*) Liefert den Tangens des angegebenen Winkels (in Bogenmaß).

TEILKETTE(*Haupt,Teil*)

Haupt = Text
Teil = Text

Findet das erste Auftreten des Teilbegriffs in der Hauptzeichenkette und liefert die Position des ersten Zeichens. Wird keine Entsprechung gefunden, ist das Ergebnis Null. Die Schreibung (groß oder klein) wird berücksichtigt.

`teilkette ("Januar","Jan")` Ergebnis: 1
`teilkette ("Januar","an")` Ergebnis: 2
`teilkette ("Januar","AN")` Ergebnis: 0

VORZ(*n*) Liefert +1 bei einem positiven, –1 bei einem negativen und 0 bei einem Null-Argument.

WENN(*Ausdruck,wahr,falsch*)
Berechnet den Wert des Ausdrucks, weist ihm den entsprechenden Wahrheitswert zu und entscheidet in Abhängigkeit davon, welches von zwei Argumenten ausgegeben wird.

Ausdruck = n
wahr = n oder Text
falsch = n oder Text

Ergibt der Ausdruck 0, d.h. falsch, kommt das Falsch-Argument zur Anwendung; jeder Ausdruck, der nicht gleich Null ist, wird als wahr interpretiert und bewirkt die Ausgabe des Wahr-Arguments. Die Argumente können numerische oder Textform haben. Beispiele:

```
wenn(A1=B1,"identisch","nicht identisch")
wenn(A1,1,0)
```

Ein Text- und ein numerisches Argument können auch gemischt werden, wie im folgenden Beispiel. Probieren Sie es aus, wenn Sie nicht ganz sicher sind, wie die Funktion sich verhält.

```
[A1] 1
[B1] 0
[C1] wenn(A1 oder B1,"eins von beiden",0)
```

WIEDERH(*Text,n*)
Füllt das aktuelle Feld mit n Kopien (Wiederholungen) des ersten Zeichens im spezifizierten Text. Beispiele:

```
wiederh("*",5)       {setzt 5 Sternchen in das aktuelle Feld}
wiederh("abc",3)     {setzt 3 a's in das Feld}
```

WTN(*n*) Gibt den Winkel (Bogenmaß) aus, dessen Tangens n ist.

ZCHN(*n*) Liefert das Zeichen, dessen Code n ist. Zeichen mit ASCII-Codes unter 32 werden nicht am Bildschirm angezeigt, werden jedoch an den Drucker gesandt (wenn der betreffende Tabellenausschnitt für Druckausgabe vorgesehen ist), vorausgesetzt, sie enthalten eine führende ASCII-Null. So bewirkt z.B. **zchn(0)+zchn(13)** die Übermittlung des ASCII-Codes für einen Wagenrücklauf als Steuerzeichen an den Drucker.

Eingabe von **zchn(160)** z.B. erzeugt ein Ä am Bildschirm.

ZEIT() Liefert als Textwert die Uhrzeit im Format "hh:mm:ss". Vorausssetzung ist das Setzen der internen System-Uhr, beschrieben bei dem Befehl SDATE von SuperBASIC.

ZLE() Gibt die Nummer der Zeile des aktuellen Feldes aus.

FEHLER

Tabellenfehler

Syntaxfehler in Formeln, z.B. eine falsche Zahl von Funktionsargumenten oder eine vereinzelte Klammer ohne Entsprechung, quittiert Abacus sofort bei der Eingabe mit einer Meldung, die aussagt, um welchen Fehler es sich handelt. Dabei bleibt die Formel weiterhin in der Eingabezeile, so daß Sie die notwendige Korrektur mit dem Zeileneditor vornehmen können.

Andere Fehler, z.B. bei Eingabe einer Formel, welche den Inhalt zweier Felder addiert, von denen das eine einen numerischen Wert und das andere eine Zeichenkette enthält, werden erst nachträglich entdeckt, d.h. bei der Kalkulation.

Bei einem Verweis auf ein leeres Feld innerhalb einer Formel nimmt Abacus an, daß es sich um einen Nullwert handelt. Dies führt unter Umständen zu einer Fehlersituation.

Bei Entdeckung eines Fehlers während der Berechnung einer Formel reagiert Abacus mit einem kurzen Fehlervermerk in der betroffenen Zelle. Setzen Sie den Cursor in das Feld und überprüfen Sie den Inhalt.

Fehlermeldungen:

Meldung	Beispiel
Fehlendes " am Ende einer Zeichenkette	`"abc" + "def`
Falsch formulierte numerische Konstante	`1.5e` (Zahl nach e fehlt)
Zahl außerhalb des Bereichs	`1.5e99`
Unzulässiges Zeichen	`12_5` (Unterstreichung statt Minus)
Alle Namen müssen für Spalten gelten	
Alle Namen müssen für Zeilen gelten	
Namenreferenzen müssen relativ sein (vgl. "Feldverweise" in diesem Kap.)	
Falsch formulierte Bereichsreferenz	
Falsch formulierte Namenreferenz	
Name ist weder Zeile noch Spalte	
Erste Namenreferenz nicht definiert	
Zweite Namenreferenz nicht definiert (Keine solche Zeichenkette oberhalb oder links in der Tabelle)	
Funktion erfordert eine Bereichsreferenz	
Unzulässiger Bereich	
Syntaxfehler	
Nicht übereinstimmende Klammern	
Keine Typenübereinstimmung	`1 + "abc"`
Falsche Zahl von Funktionsargumenten	`qwurzel(1,2)`
Zeichenkette länger als 255 Zeichen	`wiederh("*",256)`
Division durch Null	
Unzulässige Funktionsargumente	`qwurzel(-1)`
Zeichenketten-Index außerhalb des Bereichs (Index kleiner als 1 oder größer als 255, bzw. Index größer als Textumfang)	
Bezugnahme außerhalb des Bereichs (auf ein Feld außerhalb der Tabelle)	
Bezugnahme auf ein Fehler-Feld (Verweis auf ein Feld, welches einen der unten beschriebenen Fehler erzeugt)	
Speicher voll, mit WEGNEHMEN Platz machen	

Folgende Fehlervermerke können auftreten:

##TYP – Die Formel enthält einen Verweis auf ein Feld mit falschem Datentyp, d.h. numerisch statt Text oder umgekehrt.

##LANG – Die Formel enthält einen Verweis auf eine Zeichenkette mit mehr als 255 Zeichen.

##NULL – Die Formel unternimmt den Versuch einer Division durch 0.

##ARG – Die Formel enthält eine Funktion mit einem nicht zulässigen Argument, z.B. ln(-5).

##SUB – Die Formel verwendet einen Kettentrennoperator mit einem Fehler in einem oder mehreren seiner Indizes (Subskripte).

##REF – Die Formel enthält einen Verweis auf ein Feld außerhalb der Tabelle. Die Formel bewirkt die Anzeige von "FEHLER" für jeden unzulässigen Feldverweis.

##FEH – Die Formel enthält einen Verweis auf eine fehlerhafte Zelle. Dieser Fehlervermerk kann ignoriert werden, weil er mit der Richtigstellung des ursächlichen Fehlers im Bezugsfeld verschwindet.

Dateien-Fehler

Die folgenden Fehler treten im Zusammenhang mit Befehlen auf, die sich auf Dateien beziehen, also z.B. **laden** oder **dateien**.

Datei nicht vorhanden
Der eingegebene Dateiname wurde auf der Kassette in Microdrive 2 nicht gefunden.

Datei E/A unvollständig
Ein Vorgang zum Laden oder Sichern einer Datei wurde erfolgreich gestartet, läuft aber an irgendeiner Stelle nicht weiter. Dies kann ein Hinweis auf zerstörte Daten oder auf eine Beschädigung der Kassette sein.

Kann Datei nicht eröffnen
Der Versuch, die genannte Datei zu eröffnen, schlägt fehl. Gleiche Ursachen wie beim vorgenannten Fehler.

Falscher Dateityp
Der Nachname (Namenzusatz) der Datei ist nicht der von Abacus erwartete. Diese Fehlersituation tritt beispielsweise ein, wenn eine Exportdatei geladen statt importiert wird.

Unzulässiger Dateiname
Der Dateiname beginnt mit einem unzulässigen Zeichen, z.B. "3test", oder ist länger als 8 Zeichen.

Unzulässiges Import-Dateiformat
Dieser Fehler tritt beim Versuch auf, eine Datei zu importieren, die nicht mit dem **export**-Befehl erstellt wurde.

KAPITEL 1
WAS IST QL ABACUS? 1

KAPITEL 2
FANGEN SIE AN 2
QL ABACUS LADEN 2
ANZEIGEFORMAT 2
Das Fenster 3
Der Statusbereich 4
Der Steuerbereich 4
CURSOR-STEUERUNG 4
WERTE EINGEBEN 5
TEXTEINGABE 5
DIE BEFEHLE 6

KAPITEL 3
FELDER, ZEILEN, SPALTEN UND BEREICHE 7
FELDER 7
Bündig 7
Leere Felder 7
ZEILEN 8
SPALTEN 8
NAMEN 9
Namen für Zeilen und Spalten 9
Felder benennen 10
BEREICHE 11
MEHR ZU WERTEN UND TEXTEN 12

KAPITEL 4
FUNKTIONEN UND FORMELN 15
FUNKTIONEN 15
FORMELN 15
Einfache Cash-Flow-Analyse 16
AUTO-KALKULATION 17

KAPITEL 5
BEISPIELE 19
CASH-FLOW-BERECHNUNG 19
MULTIPLIKATIONS-TABELLEN 21
SCHECKHEFT-KONTROLLE 23
STANDARD-ABWEICHUNG 25
EIN HAUSHALTSBUDGET 27
BALKENDIAGRAMM MIT AUTOSKALA 29
TILGUNGSPLAN 31
Berechnung der Rückzahlungen 31
ARBEITSBLATT TILGUNGSPLAN 32
FOURIER-ANALYSE 33
Berechnung der Fourier-Transformation . . 33
Die Kosinus-Komponenten 33
Die Sinus-Komponenten 34
Das Leistungsspektrum 34
GRAFISCHE DARSTELLUNG DER FOURIER TRANSFORMATION 34
Die Transformation benutzen 35

KAPITEL 6
QL ABACUS ÜBERSICHT 36
FUNKTIONSTASTEN 36
FELDVERWEISE 36
Einzelfelder 36
Bereichsverweise 36
Zeilen- und Spaltenverweise 36
Bereichsbezeichner 36
Relative und absolute Feldadressen 36
NAMEN 37
Zeilen- und Spaltennamen 37
Feldnamen 38
FORMELN 38
Stammformeln 38
DIE BEFEHLE 39
ÄNDERN (Ä) 39
AUSDRUCK (A) 39
BÜNDIG (B) 39
DATEIEN (D) 40
ECHO (E) 40
FORMAT (F) 40
KOPIE (K) 41
LADEN (L) 41
MISCHEN (M) 42
NEUKALK (N) 42
ÖFFNUNG (Ö) 42
ORDNEN (O) 42
RASTER (R) 42
SICHERN (S) 43
TILGEN (T) 43
VERLASSEN (V) 43
WEGNEHMEN (W) 43
ZAHLEN (Z) 43
FUNKTIONEN 44
ABS 45
ANFN 45
ANFT 45
ANZAHL 45
BOG 45
BREITE 45
CODE 45
DATUM 45
DURCHSCHN 45
EXP 45
GANZZAHL 45
GNW 45
GRAD 46
IER 46
INDEX 47
KETTE 47
KOS 47
LÄNGE 47
LN 47
MAX 47
MIN 47
MONAT 47
NUMWERT 47
PI 48
QWURZEL 48
SIN 48
SP 48
SUCHEN 48
SUMME 48
TAGE 48
TAN 48
TEILKETTE 48
VORZ 48
WENN 49
WIEDERH 49
WTN 49
ZCHN 49
ZEIT 49
ZLE 49
FEHLER 49
Tabellenfehler 49
Dateien-Fehler 51

sinclair

QL

QL Archive

WICHTIG

BEVOR SIE DIE APPLIKATIONEN BENUTZEN, LESEN SIE BITTE DIE NACHSTEHENDE BESCHREIBUNG, UM ARBEITSKOPIEN ZU ERSTELLEN.

ARBEITSKOPIEN

Fertigen Sie bitte immer mindestens eine Arbeitskopie an, bevor Sie eines der vier QL Programme benutzen. Arbeiten Sie dann immer nur mit der Arbeitskopie. Bewahren Sie die Originalkassetten mit den Programmen sicher auf. Benutzen Sie die Originalkassetten nur, um bei Bedarf Kopien anzufertigen. So schützen Sie sich vor dem Verlust Ihrer Programme.

Bei allen magnetischen Speichermedien, also auch bei Microdrive-Kassetten, kann bei Gebrauch Datenverlust entstehen. Daher sollten Sie alle Programme und Daten, die Sie noch benötigen, mindestens zweimal speichern. Fertigen Sie sich deshalb immer Sicherungskopien an. So schützen Sie sich vor Datenverlust. Immer, wenn Sie einer Kassette Daten oder Programme hinzufügen, sollten Sie das auch bei der Sicherungskopie tun. Wenn Sie das versäumen, laufen Sie Gefahr, alles, was Sie seit der letzten Datensicherung hinzugefügt haben, zu verlieren. Hinweise zur Anfertigung von Sicherungskopien finden Sie im Abschnitt *Information*.

by Dick de Grandis-Harrison *(Psion Limited)*

KAPITEL 1

WAS IST QL ARCHIVE?

QL Archive ist ein Datenbankprogramm, welches Ihnen ermöglicht, beliebige Informationen systematisch zu archivieren, zu verwalten, zu sortieren und abzufragen. Sie haben volle Entscheidungsfreiheit, wie Sie Ihre Datei aufbauen und auf die Daten zugreifen wollen.

In kürzester Zeit werden Sie entdecken, wie man mit Archive einfache Dateien, etwa Adreßlisten und Kundenkarteien, erstellt. Und wenn Sie das beherrschen, ist es kein großer Schritt mehr zur Entwicklung komplizierterer relationaler Dateien, wo der Informationsvorrat von mehreren Dateien geteilt und gemeinsam verwaltet wird, etwa zwischen einer Einkaufs– und einer Lager-Datei.

Sie können sich die Informationen entweder mit der standardmäßig von Archive vorgesehenen Bildschirmmaske ausgeben lassen oder selbst eine Maske definieren. Selbstverständlich ist auch Ausgabe über einen Drucker möglich, z.B. als Formulare, Etiketten oder Berichte.

Eines der wesentlichen Leistungsmerkmale von Archive ist seine Befehlsstruktur. Mit den Archive-Befehlen können Sie eine einmal angelegte Datei nach bestimmten Datensätzen durchsuchen und nach gewissen Kriterien ordnen, sortieren und anzeigen.

Alle Befehle zusammen bilden eine mächtige Datenbank-Sprache, die eine gewisse Ähnlichkeit mit der Programmiersprache SuperBASIC aufweist. Sie erlaubt das Schreiben komplexer Programme für spezialisierte Anwendungen.

Beim Arbeiten mit Archive hält Sie das System ständig auf dem laufenden, welche Möglichkeiten zur Auswahl stehen, welche Schritte zu unternehmen sind und welche Eingabe erwartet wird. Und wenn Sie einmal nicht weiterwissen, hilft ein Tastendruck auf **F1**. Die Hilfe-Datei ist hierarchisch organisiert und die Auskünfte situationsbezogen.

Die eigentliche Leistungsfähigkeit von Archive zeigt sich beim Schreiben eigener Prozeduren in der Befehlssprache. Archive bietet dem Benutzer die Möglichkeit, Prozeduren für seine individuellen Anforderungen zu definieren und zu benennen und diese dann genau wie die eingebauten, systeminternen Befehle zu verwenden.

Zum Schreiben und Überarbeiten solcher Prozeduren gibt es neben dem ständig verfügbaren *Eingabezeilen-Editor* noch einen speziellen *Programm-Editor*. Auf diese Weise ist eine optimale Unterstützung garantiert.

Die Dateien arbeiten mit variablen Feldlängen und Datensätzen. Das garantiert nicht nur die bestmögliche Ausnutzung der Arbeits– und Kassettenspeicherkapazität, sondern vereinfacht auch das Anlegen der Dateien ganz erheblich, weil Sie die Datenlänge nicht im voraus festlegen müssen.

Das vorliegende Handbuch enthält eine Reihe von Übungsbeispielen. Wir empfehlen Ihnen, diese auszuprobieren, um sich mit Archive anzufreunden und seine vielfältigen Einsatzgebiete kennenzulernen. Außerdem werden Sie darunter auch Mehrzweck–Prozeduren finden, die Sie vielleicht für eigene Anwendungen übernehmen wollen.

Falls Sie irgendwann nicht sicher sind, wie es weitergeht, holen Sie sich mit der Funktionstaste **F1** Hilfe. Vergessen Sie auch nicht, daß irrtümlich angeforderte, noch nicht beendete Operationen, etwa eine falsche numerische Eingabe oder ein versehentlich gedrückter Befehl, mit **ESC** rückgängig gemacht werden können.

Archive wurde im Hinblick auf größtmögliche Flexibilität entwickelt. Dies bedingte eine gewisse Sparsamkeit bei der Bedienerführung, die im Vergleich zu den anderen QL-Programmen etwas knapper ist. Wenn Sie sich zum erstenmal mit Computern und Programmieren beschäftigen, sollten Sie erst den SuperBASIC Anfänger-Kurs in diesem Handbuch studieren, bevor Sie sich ans Schreiben von Archive-Programmen wagen.

KAPITEL 2
FANGEN SIE AN

QL ARCHIVE LADEN

Laden Sie QL Archive gemäß der Anleitung in der Einführung zu den QL-Programmen. Nach abgeschlossenem Ladevorgang meldet sich das System mit der Anzeige:

```
LADEN DER QL ARCHIVE
Datenverwaltung
Version x.y
Copyright ©1984 PSION Ltd.
Alle Rechte vorbehalten
```

wobei x.y die Versions-Nummer bedeutet, z.B. 2.23.

Das Programm schiebt vor dem Start eine kurze Pause ein.

Die Hilfe-Datei wird nicht zusammen mit dem eigentlichen Programm in den Arbeitsspeicher geladen, sondern erst bei Bedarf von der Kassette eingelesen. **Aus diesem Grund sollten Sie die Archive–Kassette im Microdrive 1 belassen, wenn Sie vorhaben, zu irgendeinem Zeitpunkt die Hilfe aufzurufen.**

ANZEIGEFORMAT

Nach dem Ladevorgang sollte auf Ihrem Bildschirm eine Maske wie in Abbildung 2.1 erscheinen. Das ist die *Hauptanzeige*.

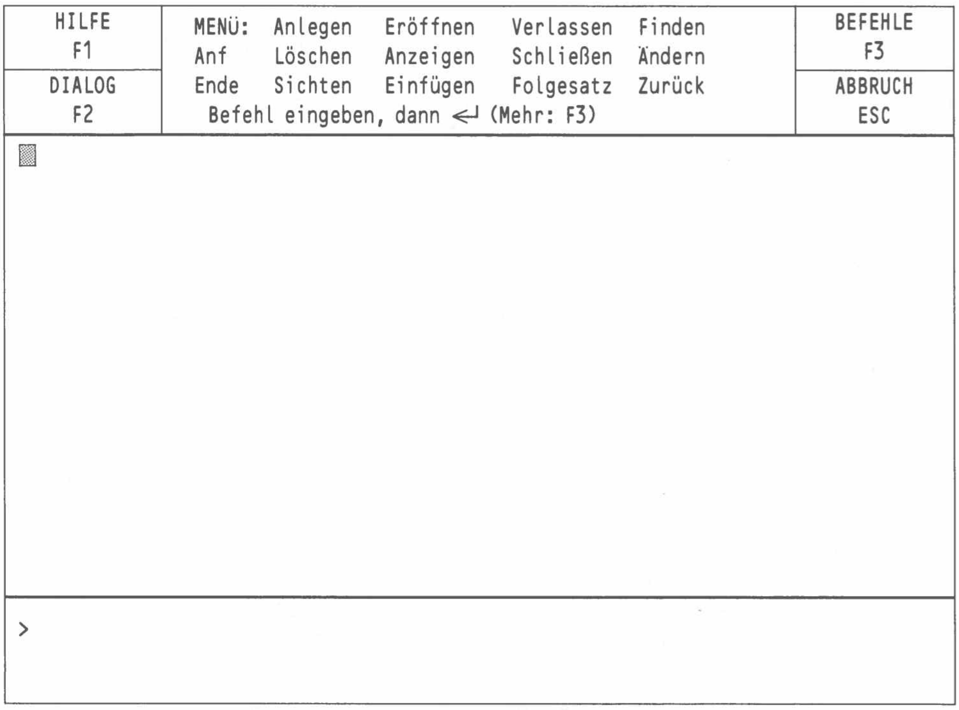


Abbildung 2.1. Die Hauptanzeige auf einem Monitor (80 Zeichen pro Zeile)

Bei Verwendung eines Heimfernsehers ist der Bildschirmaufbau etwas anders, weil dort statt 80 nur 64 Zeichen je Zeile (Spalten) angezeigt werden.

Die Bildschirmfläche unterteilt sich in die drei Bereiche: Anzeigebereich, Arbeitsbereich und Steuerbereich.

Anzeige– und Arbeitsbereich

Der Anzeigebereich ist, wie der Name sagt, für die Anzeige der Archive-Informationen reserviert.

Der Arbeitsbereich belegt einen vierzeiligen Streifen am unteren Rand des Bildschirms. Hier werden die Befehle angezeigt, die Sie eingeben, und etwaige Fehlermeldungen.

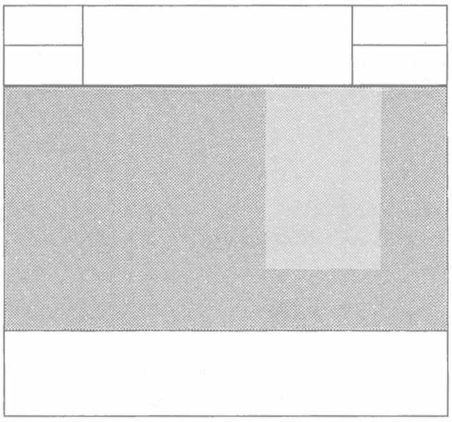


Abbildung 2.2 Der Anzeigebereich

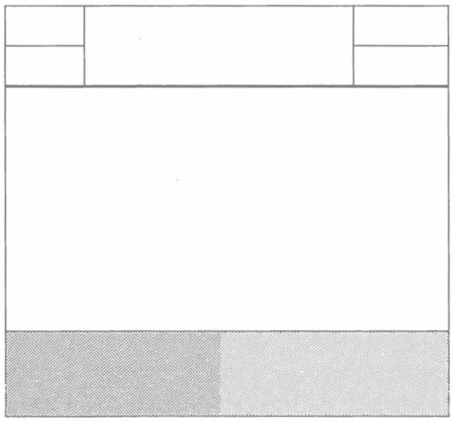


Abbildung 2.3 Der Arbeitsbereich

Diese beiden Bereiche sind eng miteinander verknüpft, insofern, als die Resultate der im Arbeitsbereich eingegebenen Befehle im Anzeigebereich erscheinen. Geben Sie einmal folgendes Programm ein:

```
setzen x=14:solange x> 0:zeigen x:
setzen x=x- 1:endesolange [ENTER]
```

Der Programmtext wird in der ersten Zeile des Arbeitsbereichs ausgegeben. Nach Drücken von **ENTER** erscheint im Anzeigebereich ganz links eine Kolonne von Zahlen, die in der oberen Zeile mit 14 beginnt und bei 1 endet. Die untere Zeile des Anzeigebereichs bleibt leer, mit Ausnahme des roten Cursors, der die nächste Anzeigeposition markiert. Insgesamt umfaßt also der Anzeigebereich 15 Zeilen.

Mit dem Befehl

```
leeren [ENTER]
```

leeren Sie den gesamten Anzeigebereich.

Der Steuerbereich

Der Steuerbereich ist die eingerahmte Leiste am Kopfrand. Sie informiert über die allgemeinen Funktionen: HILFE **(F1)**, Dialogmeldungen ein/aus **(F2)**, Abbruch nicht beendeter Operationen **(ESC)** und Aufruf des Befehlsmenüs **(F3)**.

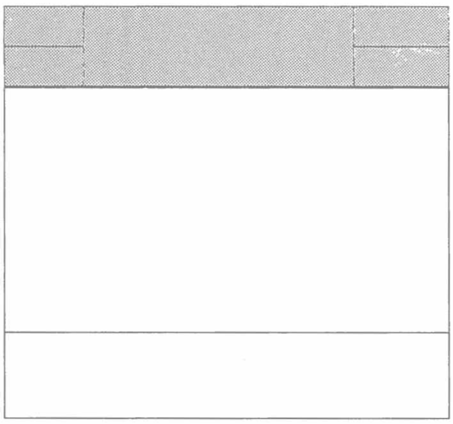


Abbildung 2.4 Der Steuerbereich

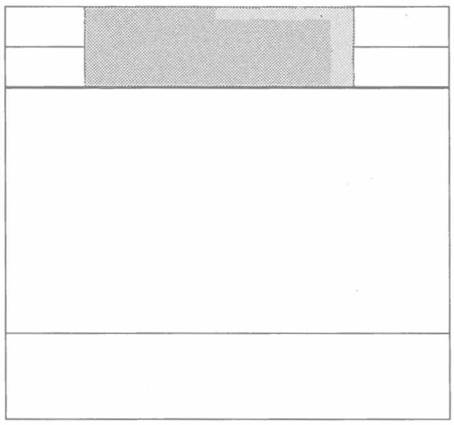


Abbildung 2.5 Die Archive-Befehle

UMGANG MIT BEFEHLEN

Die Archive-Befehle bilden eine richtige Programmiersprache, und ihre Namen müssen ganz ausgeschrieben werden. Wenn Ihnen das etwas umständlich erscheint, können wir Ihnen jetzt schon sagen, daß es möglich ist, mit Hilfe von Prozeduren einzelne Tasten mit Befehlen zu belegen, so daß ein Tastendruck ausreicht, um einen Befehl einzugeben. Doch davon später.

Die Befehle verteilen sich auf vier verschiedene Menükarten, die mit **F3** der Reihe nach aufgerufen werden. Diese Befehle werden durch Eingabe ihres Namens und **ENTER** gestartet; manche fordern noch zusätzliche Informationen an.

Es können beliebige Befehle aus dem Archive-Vorrat aufgerufen werden, auch solche, die zu einem anderen als dem aktuellen Menü gehören.

Ein Befehl kann beliebig in Groß– und Kleinschreibung eingegeben werden. Meistens ist es bequemer, Befehle nur in Kleinbuchstaben einzugeben. Deshalb werden Befehle im Text dieses Handbuches normalerweise in Kleinschreibung aufgeführt.

DER MODUS-BEFEHL

Mit dem **modus**-Befehl können alle drei Bildschirmbereiche zu einem sog. Vollbild zusammengefaßt werden. Eingabe von **modus 0** hat dieselbe Wirkung. Probieren Sie's mit

```
modus [ENTER]
```

worauf der gesamte Bildschirm für Tastatureingabe und Befehls– oder Programmausgabe gemeinsam zur Verfügung steht. Der **modus**-Befehl mit dem Wert 1 verwandelt die Fläche wieder in drei Teilbilder.

Der gleiche Befehl dient auch dazu, die Anzahl der Buchstaben zu ändern, die in einer Zeile angezeigt werden. Zu diesem Zweck fügen Sie einen zweiten Parameter hinzu, der durch ein Komma vom ersten zu trennen ist. In Frage kommen die Zahlen 4, 6 oder 8, welche 40, 64 bzw. 80 Buchstaben pro Zeile entsprechen. Geben Sie

```
modus 0,4 [ENTER]
```

ein, um die Anzeige auf Vollbild und das Format auf 40 Spalten zu ändern.

Experimentieren Sie mit verschiedenen Kombinationen und verfolgen Sie den Effekt auf dem Bildschirm. Kehren Sie am Schluß auf eine Anzeige mit drei getrennten Bereichen zurück und wählen Sie die Anzeigebreite, die ein scharfes Bild erzeugt.

KAPITEL 3

QL ARCHIVE DATEIEN

DATEIEN, DATENSÄTZE UND FELDER

Eine Archive-Datei kann man gut mit einer Kartei vergleichen. Unter einer herkömmlichen Kartei versteht man einen Zettelkasten, der Karten mit einer Reihe von Angaben enthält, die man schnell nachschlagen kann. Das funktioniert nur, wenn die Informationen auf den einzelnen Karten systematisch in der gleichen Rubrik und Position geführt werden.

Gehen wir von einer Namen-Adressen-Kartei aus. Normalerweise wird der Name ganz oben geschrieben und darunter die Anschrift und gegebenenfalls die Telefonnummer. Es wäre etwas entnervend, wenn der Name bei manchen Karten oben und bei anderen unten vermerkt wäre. Die Idee ist ja doch, daß man beim Durchsehen der oberen Zeile ganz schnell die richtige Karte herausgreifen kann.

Wenn man zwei Zettelsammlungen hat, eine mit Kundennamen und eine mit Lagerposten, dann empfiehlt es sich, die beiden getrennt aufzubewahren und sie nicht in einem Zettelkasten zu mischen. Man würde folglich eine Kartei mit den Namen "Kunden" und eine zweite mit dem Namen "Lager" einrichten.

Das Prinzip einer solchen Kartei enthält bereits die meisten Ideen, die zum Verständnis einer Archive-Datei notwendig sind. Der Zettelkasten ist die Datei, der zur eindeutigen Identifikation ein Name zugewiesen wird. Die einzelnen Karteikarten sind die *Datensätze* (englisch: records). Eine Datei ist also einfach eine Sammlung von zusammengehörigen Datensätzen.

Genau wie die Karteikarten sind auch die Datensätze nach einem Schema, einer regelmäßigen Struktur, organisiert, so daß z.B. die Telefonnummern immer an derselben Stelle auf der Karte eingetragen werden. Bei den Datensätzen ist jedem einzelnen Eintrag eine bestimmte Rubrik vorbehalten, die man *Feld* nennt. Jeder Datensatz in der oben erwähnten Kundendatei würde ein Namenfeld, ein Adreßfeld, ein Telefonfeld usw. enthalten.

Wenn die Sache damit zu Ende wäre, könnte man zu Recht fragen, ob es sich denn überhaupt lohne, den guten alten Zettelkasten durch eine Archive-Datei zu ersetzen. Aber wir sind ja erst am Anfang: computerisierte Karteien haben eine Menge von Vorteilen, die Sie nach und nach kennernlernen werden.

Kundenkarteien sind im allgemeinen alphabetisch geordnet. Es ist also nicht schwer, die Anschrift und die Telefonnummer von Herrn Baer ausfinding zu machen. Nehmen wir aber einmal an, Sie wollten plötzlich ein Rundschreiben an alle Ihre Kunden schicken, die seit einem halben Jahr nichts mehr bei Ihnen bestellt haben. Das Heraussuchen der entsprechenden Angaben und das Zusammenstellen einer vollständigen Liste wäre ganz schön mühsam und zeitaufwendig. Diese komplexe Arbeit erledigt QL Archive im Handumdrehen aufgrund von ein paar Befehlen, und auf Ihren Wunsch druckt es gleichzeitig auch schon die Etiketten aus.

Schon an diesem Beispiel wird deutlich, daß Sie sich eine Menge Zeit und Mühe sparen, wenn Sie die Verwaltung und Archivierung Ihrer Daten QL Archive überlassen.

KAPITEL 4

EINBLICK IN DATEIEN

Die beste Methode, Archive kennenzulernen, ist anhand der Demonstrationsdatei **Staaten**, welche sich auf der Utilities-Kassette befindet. Sie enthält Informationen zu verschiedenen Ländern, den Kontinent, die Hauptstadt, die Währung, die Sprache(n), die Bevölkerung, die Fläche und das Bruttosozialprodukt pro Einwohner.

Die meisten Beispiele in den Kapiteln 4 und 5 beziehen sich auf diese Datei. Bevor Sie damit arbeiten, sollten Sie vom Original eine Reservekopie erstellen. Dazu gehen Sie wie folgt vor:

Nach dem Laden von Archive nehmen Sie die Archive-Kassette vorübergehend aus dem Microdrive 1 und legen die Utilities-Kassette ein. Legen Sie in Microdrive 2 eine formatierte Kassette ein und geben Sie ein:

```
reserve [ENTER]
mdv1_Staaten_dbf [ENTER]
mdv2_Staaten_dbf [ENTER]
```

Warten Sie, bis beide Microdrives nicht mehr laufen. Da die Datei recht lang ist, kann das eine Weile dauern. Verwenden Sie jetzt für die Übungen die in Microdrive 2 befindliche Kopie.

Wenn Sie die Reservekopie erstellt haben, können Sie die Utilities-Kassette wieder aus dem Microdrive 1 herausnehmen und die Archive-Kassette einlegen.

Hinweis: Jeder Archive-Befehl muß durch Drücken der **ENTER**-Taste abgeschlossen werden. Erst dann wird er vom System erkannt und verarbeitet. Im folgenden werden wir dies nicht mehr nach jedem Befehl ausdrücklich angeben.

Der **sichten**-Befehl eröffnet eine Datei, jedoch nur zum Lesen und nicht etwa zum Ändern oder Ergänzen. Er ist weniger riskant als der **eröffnen**-Befehl, weil keine versehentlichen Änderungen passieren können. Zum Besichtigen der Kopie von "Staaten" auf Microdrive 2 lautet der Befehl:

```
sichten "Staaten"
```

DATENSATZ ANZEIGEN

Für eine Anzeige des ersten Datensatzes diese zwei Befehle eingeben:

```
anf
anzeigen
```

Denken Sie daran, jeden einzelnen Befehl mit **ENTER** abzuschließen. Daraufhin wird der erste Datensatz angezeigt.

Es sei darauf hingewiesen, daß der Name in der ersten Zeile der logische Name der Datei ist, der von Archive bei Einzeldateien automatisch mit "Haupt" bezeichnet wird. Logische Dateinamen werden im allgemeinen beim Arbeiten mit mehr als einer Datei gebraucht. Eine Beschreibung folgt später.

WEITERE DATENSÄTZE

Nach Durchsehen des ersten Datensatzes wollen Sie vielleicht zum nächsten übergehen. Zu diesem Zweck geben Sie ein:

```
folgesatz
```

worauf der folgende Datensatz angezeigt wird. Nach Anforderung von **anzeigen** wird der Anzeigebereich bei Eingabe von einzelnen Befehlen laufend auf den aktuellen Datensatz geändert. Mit **folgesatz** können Sie die ganze Datei, Satz um Satz, durchschreiten, bis zum letzten Datensatz, über den nicht hinausgegangen werden kann.

Daneben gibt es noch drei andere, verwandte Befehle, die zur Wahl eines bestimmten Datensatzes dienen:

zurück	zur Anzeige des vorangehenden Datensatzes
anf	zur Anzeige des ersten Datensatzes
ende	zur Anzeige des letzten Datensatzes in der Datei.

Probieren Sie diese Befehle aus und lassen Sie sich beliebige Datensätze anzeigen. Beachten Sie dabei, daß die vier Befehle **anf, ende, folgesatz** und **zurück** an sich noch keine Anzeige bewirken, sondern lediglich den Sprung von einem Datensatz auf einen anderen, unabhängig davon, ob gleichzeitig der **anzeigen**-Befehl wirksam ist.

DURCHSUCHEN EINER DATEI

Finden

Der erste und einfachste Suchbefehl ist **finden**. Er durchsucht die Datei von Anfang an auf die erste Entsprechung des Suchpuments in einem der Textfelder. Beispiel:

```
finden "Afrika"
```

Nach Drücken von **ENTER** folgt eine kurze Pause und danach die Anzeige des ersten Datensatzes, der ein Feld mit dem Wort "afrika" enthält. Diese Suchmethode berücksichtigt die Schreibung nicht; "AFRIKA", "afrika" und "Afrika" werden gleichermaßen entdeckt.

Weiter

Wenn der erste gefundene Datensatz nicht der gewünschte ist, kann bis zur nächsten Entsprechung weitergesucht werden – mit dem Befehl:

```
weiter
```

Der **weiter**-Befehl fährt mit der Suche fort, bis er in einem der folgenden Datensätze wieder ein Feld findet, das den Suchbegriff enthält.

Falls Sie den letzten Befehl mehrmals wiederholen müssen, bevor der richtige Datensatz aufgefunden wird, können Sie Archive durch Drücken von **F5** veranlassen, den Befehl in die Befehlszeile zu holen. **ENTER** aktiviert dann diesen Befehl von neuem.

Suchen

Eine andere Methode zum Auffinden eines bestimmten Datensatzes ist **suchen**. Hier haben Sie die Möglichkeit, den Inhalt eines oder mehrerer Datenfelder als Suchbedingung anzugeben. Die Eingabe von

```
suchen Kontinent$="EUROPA" und Sprachen$="FRANZÖSISCH"
```

z.B. würde den ersten Datensatz in der Datei heraussuchen, der diese beiden Bedingungen erfüllt. Es muß der volle Name des Feldes angegeben werden.

Im Gegensatz zu **finden**, testet **suchen** nur die spezifizierten Felder und macht eine Unterscheidung zwischen Groß- und Kleinschreibung, es sei denn, Sie geben die Funktion **groß()** bzw. **klein()** ein, um eine von Groß- und Kleinschreibung unabhängige Suche durchzuführen, etwa:

```
suchen klein(kontinent$)="europa"
```

Auch hier dient **weiter** zur Fortsetzung der Suche.

Wählen

In vielen Fällen werden Sie sich nur eine Untermenge von Datensätzen aus einer Datei anschauen wollen, z.B. nur die europäischen Länder. Zu diesem Zweck gibt es den **wählen**-Befehl, der lediglich die Datensätze herausgreift, die ein bestimmtes Merkmal aufweisen, d.h. eine spezifische Bedingung erfüllen, und sie zu einer Unterdatenbank vereinigt. Probieren Sie dies mit der "Staaten"-Datei aus, wobei Sie erst mit

```
zeigen anzahl()
```

eine Zählung aller Datensätze vornehmen. Schreiben Sie anschließend:

```
wählen Kontinent$="EUROPA"
zeigen anzahl()
```

Daraus ersehen Sie die Zahl der ausgewählten Datensätze. Die beim Selektionsvorgang herausgefallenen Datensätze befinden sich nach wie vor im Arbeitsspeicher und können jederzeit mit dem Befehl **rücksetzen** wiederhergestellt werden. Geben Sie also den Befehl

```
rücksetzen
```

ein und lassen Sie sich nochmals den Wert von **anzahl()** anzeigen, um sicherzustellen, daß die Datei wieder ihren ursprünglichen Umfang hat.

Die Eingabe von **zeigen** über die Tastatur löscht die etwa auf dem Bild-Schirm vorhandene Anzeige eines Datensatzes. Der Grund liegt darin, daß **anzeigen** und **zeigen** überlappende Bildschirmbereiche verwenden. Nach Gebrauch von **zeigen** stellen Sie durch Eingabe von **anzeigen** die Anzeige wieder her.

DATEIEN SORTIEREN

Nicht immer sind die Datensätze schon so geordnet, wie es gerade am günstigsten ist. Sortieren kann man nach dem Inhalt von Feldern. **Bei der alphabetischen Sortierung werden nur die ersten acht Buchstaben berücksichtigt.**

Beispielsweise sollen die "Staaten"-Datensätze alphabetisch nach ihren Hauptstädten sortiert werden. Zu diesem Zweck kann der **ordnen**-Befehl wie folgt eingesetzt werden:

```
ordnen Hauptstadt$;a
```

wobei das "a" nach dem Semikolon für "aufsteigend" steht. Eine umgekehrte, absteigende Folge (z bis a) wird mit dem Parameter "d" anstelle von "a" erzeugt. Das Feld Hauptstadt$ wird damit zum *Sortierschlüssel* oder *Ordnungsbegriff* für die Datei. Ein solcher Sortierschlüssel kann aus bis zu vier Feldern bestehen. Diese werden dem **ordnen**-Befehl in Form eines Parameters angehängt, jedes mit der Zusatzinformation "a" oder "d" zur Festlegung der Sortierrichtung. Der folgende Befehl sortiert die Datei in absteigender Folge nach Bevölkerungszahl und in aufsteigender Reihenfolge nach Hauptstadt:

```
ordnen Bev;d,Hauptstadt$;a
```

Achten Sie darauf, jeweils das Feld und den Zusatz "a" bzw. "d" durch Semikolon zu trennen und als Trennzeichen zwischen solchen Paaren ein Komma zu setzen.

Bei aus mehr als einem Feld bestehenden Sortierschlüsseln werden die Datensätze zunächst nach dem Inhalt des ersten Feldes aus der Liste geordnet. Bei zwei oder mehreren Datensätzen mit demselben Inhalt im ersten Feld wird die Reihenfolge aufgrund des nächsten Feldes ermittelt. Besteht dort ebenfalls Übereinstimmung, wird der Inhalt des nächsten Feldes herangezogen, usw.

ORTEN

In einer sortierten (geordneten) Datei dient der **orten**-Befehl dazu, einen bestimmten Datensatz herauszusuchen. Er ermittelt den ersten Datensatz, dessen erstes Sortierfeld größer oder gleich dem gegebenen Ausdruck ist, und macht diesen zum aktuellen Datensatz.

Nach Sortieren der "Staaten"-Datei in der oben beschriebenen Weise ermittelt der Befehl

```
orten 100
```

das erste Land in der sortierten Datei, das eine Bevölkerung von 100 Mio. aufweist. Wenn kein solches Land existiert, sucht Archive nach dem ersten Land mit einer Bevölkerung unter 100 Mio. (davon ausgehend, daß die Datei in absteigender Reihenfolge sortiert wurde).

Orten wird gefolgt von einem *Ausdruck*, der alphabetisch (Text) oder numerisch (Zahlen) sein kann, unter der Voraussetzung, daß er vom Typ her mit dem Sortierfeld übereinstimmt, nach welchem die Datei geordnet wurde. (Vgl. Kapitel 12: QL-Archive Übersicht).

Mit **orten** kann ein Datensatz im Hinblick auf mehrere Sortierfelder ermittelt werden, indem der Befehl mit Mehrfachausdrücken verwendet wird, die durch Kommas voneinander getrennt sind, z.B.

```
setzen a=100
setzen b$="D"
orten a,b$
```

womit das erste Land mit 100 Mio. Einwohnern oder weniger gefunden wird, dessen Hauptstadt entweder mit "D" oder einem späteren Buchstaben im Alphabet beginnt. In unserem Beispiel würde Archive Bangladesh mit einer Bevölkerungszahl von 87 Mio. und der Hauptstadt Dacca heraussuchen.

Die einzige Einschränkung bei der Anzahl von Ausdrücken im Zusammenhang mit **orten** ist die Zahl der Felder, die beim Sortieren der Datei mit berücksichtigt wurden.

Weiter funktioniert nicht nach **orten**. Eine wiederholte Verwendung von **orten** mit derselben Bedingung liefert immer wieder denselben Datensatz.

Orten ist die schnellste Methode zum Aufsuchen eines Datensatzes innerhalb einer umfangreichen, sortierten Datei. Unter Umständen sind weitere Prüfungen notwendig, um sicherzustellen, daß es sich um den gewünschten Datensatz handelt.

DATEI SCHLIESSEN

Wenn Sie die Besichtigung einer Datei beendet haben, muß Archive informiert werden. Dazu gibt es einmal den Befehl

```
schließen
```

der lediglich auf Dateien wirkt, hingegen Programme und Bildschirmmaske bestehen läßt. Daneben gibt es einen zweiten Befehl

```
neu
```

der alle Dateien schließt und die Daten und den Anzeigebereich löscht, d.h. Archive in seinen Anfangszustand versetzt, wie er nach dem Laden besteht.

Um die Arbeit mit Archive zu beenden und auf SuperBASIC umzusteigen, können Sie **verlassen** verwenden. Dieser Befehl schließt automatisch alle Dateien, bevor er Archive verläßt.

Denken Sie daran, niemals eine Kassette aus dem Microdrive herauszunehmen, solange sie noch offene Dateien enthält.

KAPITEL 5
DATEIEN BEARBEITEN

Bevor Sie damit beginnen, die Beispiele in diesem Kapitel einzugeben, versetzen Sie Archive mit dem Befehl **neu** in den Anfangszustand.

Der **eröffnen**-Befehl eröffnet eine Datei zum Lesen und zum Schreiben.

Anders als beim Eröffnungsbefehl **sichten** ist die Datei damit nicht nur für Leseoperationen zugänglich, sondern ebenso zur Bearbeitung und Veränderung. Wenn Sie jetzt Text hinzufügen, löschen oder korrigieren, dann ändert sich der Inhalt der "Staaten"-Datei, mit der Sie arbeiten. Geben Sie ein:

```
eröffnen "Staaten"
```

Es muß betont werden, daß die im folgenden beschriebenen Befehle zur Bearbeitung von Dateien immer die Eröffnung mit **eröffnen** voraussetzen. Wurde stattdessen **sichten** verwendet, reagiert das System mit einer Fehlermeldung, sobald ein Befehl eingegeben wird, der die Datei in irgendeiner Weise zu verändern versucht, d.h. einer der in diesem Kapitel neu eingeführten Befehle.

Lassen Sie sich den ersten Datensatz anzeigen:

```
anf
anzeigen
```

EINFÜGEN

Der **einfügen**-Befehl dient dazu, eine oder mehrere Datensätze in die aktuelle Datei aufzunehmen. Bei Verwendung von **einfügen** werden Sie aufgefordert, den Inhalt der einzelnen Felder des neuen Datensatzes einzutragen. Geben Sie den Befehl

```
einfügen
```

ein, worauf im Anzeigebereich diese Aufstellung erscheint:

```
Logischer Name      : Haupt
Land$               :
Kontinent$          :
Hauptstadt$         :
Währung$            :
Sprachen$           :
Bev                 :
Fläche              :
BSP                 :
```

Jetzt können Sie in jedem Feld die entsprechende Eintragung vornehmen. Der Sprung von einem Feld zum nächsten erfolgt mit **ENTER** oder **TAB**; ein Rückwärtssprung mit **SHIFT** plus **TAB**. Sie können die Felder beliebig korrigieren und überschreiben, bevor Sie die endgültige Fassung durch Drücken von **F5** oder **ENTER** im letzten Feld in die Datei übernehmen. Zum Verlassen von **einfügen** **F4** drücken.

Versuchen Sie es mit diesem neuen Datensatz:

```
SCHOTTLAND          [TAB]
EUROPA              [TAB]
EDINBURGH           [TAB]
PFUND STERLING      [TAB]
ENGLISCH            [TAB]
10                  [TAB]
30                  [TAB]
50                  [TAB]
```

worauf Sie im Anzeigebereich Ihre Eingabe überprüfen können:

```
Logischer Name      : Haupt
Lande$              : SCHOTTLAND
Kontinent$          : EUROPA
Hauptstadt$         : EDINBURGH
Währung$            : PFUND STERLING
Sprachen$           : ENGLISCH
Bev                 : 10
Fläche              : 30
BSP                 : 50
```

Wenn Sie mit den Angaben zufrieden sind, drücken Sie **F5** zum Einfügen des neuen Datensatzes in die Datei, worauf die Feldinhalte vom Bildschirm verschwinden, damit Sie einen neuen Datensatz anlegen können. Zum Verlassen des **einfügen**-Befehls dient F4.

Die Eingabe kann auch feldweise erfolgen, indem Sie nach Ausfüllen eines Feldes jeweils**ENTER** drücken (statt **TAB**). Nach Eingabe des letzten Feldinhalts durch **ENTER** wird der Datensatz automatisch in die Datei übernommen.

Handelt es sich um eine sortierte Datei, wird der neue Datensatz an der richtigen Stelle eingefügt.

LÖSCHEN

Dieser Befehl löscht den aktuellen Datensatz aus der Datei. Vergewissern Sie sich, daß dies der Datensatz ist, der gelöscht werden soll, und geben Sie dann den Befehl

```
löschen
```

ein.

DATENSÄTZE BEARBEITEN

Es ist ohne weiteres möglich, bereits bestehende Datensätze zu ändern, z.B. den Inhalt eines oder mehrerer Felder zu korrigieren oder auf den neuesten Stand zu bringen. Dazu gibt es zwei Methoden:

Ändern

Suchen Sie den betreffenden Datensatz mit Hilfe von **anzeigen** und **finden** und schreiben Sie dann **ändern**. Der Befehl **ändern** funktioniert genau wie **einfügen**, nur daß in jedem Feld der ursprüngliche Inhalt angezeigt wird. Felder, die unverändert belassen werden sollen, werden einfach mit **TAB** oder **ENTER** übersprungen. Alte Einträge können durch Überschreiben ersetzt oder mit Hilfe der Cursorpfeiltasten korrigiert werden. Drücken von **F5** bewirkt, daß der überarbeitete Datensatz anstelle des alten in die Datei übernommen wird.

Wie bei **einfügen** kommt auch hier der Austausch automatisch zustande, wenn Sie nach dem letzten Feld im Datensatz **ENTER** drücken.

Ersetzen

Suchen Sie den betreffenden Datensatz und nehmen Sie die gewünschten Änderungen an den einzelnen Feldern vor. Der neue Datensatz übernimmt die Stelle des ursprünglichen, sobald Sie den **ersetzen**–Befehl eingeben.

Nehmen wir an, Sie sind der Meinung, Island gehöre eigentlich zu Europa und nicht in die Arktis. Suchen Sie den entsprechenden Datensatz heraus:

```
finden "Island": anzeigen
```

Verwenden Sie zur Änderung des Inhalts im Feld Kontinent$ den **setzen**-Befehl:

```
setzen Kontinent$ = "Europa"
```

und veranlassen Sie die Übernahme dieser Korrektur in den Datensatz durch Eingabe von **ersetzen**.

Beide der beschriebenen Methoden plazieren den neuen Datensatz an die richtige Position, wenn es sich um eine geordnete Datei handelt. Andernfalls wird der Datensatz an einer unbestimmten Stelle eingefügt.

Der **ändern**-Befehl ist einfacher im Gebrauch, bezieht sich jedoch immer nur auf den aktuellen Datensatz. Im Gegensatz dazu ist **ersetzen** sehr praktisch, wenn mit Mehrfach-Dateien gearbeitet wird.

Vor Ausschalten des Computers ist die Datei unbedingt mit **schließen**, **neu** oder **verlassen** zu schließen.

DATEIEN SCHLIESSEN

Nach Abschluß der Änderungen und Korrekturen muß die Datei wieder geschlossen werden – mit **schließen** oder **neu** –,um sicherzustellen, daß die Datei in ihrem neuesten Zustand gespeichert wird.

Wenn diese Vorschrift nicht beachtet und der Computer z.B. einfach ausgeschaltet wird, kann der Inhalt der Datei Schaden nehmen. Die Datei kann von Archive meist nicht mehr verarbeitet werden. **Vergewissern Sie sich vor Entnahme einer Kassette aus dem Microdrive immer, daß alle Dateien geschlossen wurden. Schalten Sie niemals den Computer aus, bevor Sie nicht alle eröffneten Dateien geschlossen und die Kassetten aus den Microdrives entfernt haben.**

KAPITEL 6
DATEIEN ANLEGEN

Bei den bisherigen Beispielen haben wir auf eine Datei zurückgegriffen, die Archive speziell für diesen Zweck zur Verfügung stellt. In diesem Kapitel wollen wir Ihnen zeigen, wie Sie selbst Ihre eigenen Dateien anlegen und ihnen Namen zuweisen.

Zum Leeren des Arbeitsspeichers und zum Schließen etwaiger offener Dateien geben Sie zunächst den **neu**-Befehl ein. Vergewissern Sie sich, daß Sie in Microdrive 2 eine formatierte Kassette haben, auf der Sie eine eigene Datei erstellen können.

Nehmen wir an, Sie wollen mit Hilfe von Archive einen Katalog Ihrer Bücher anlegen. Zu diesem Zweck brauchen Sie eine neue Datei mit dem Titel "Bücher". Als erstes müssen Sie sich überlegen, welche Informationen sie enthalten soll, anders ausgedrückt, welche Felder Sie den einzelnen Datensätzen zuweisen wollen. Im vorliegenden Beispiel müßten auf jeden Fall der Autor, der Buchtitel und das Sachgebiet verzeichnet werden und auf Wunsch weitere Details, z.B. der Verlag, das Erscheinungsjahr, der Standort im Bücherregal, eine Kurzbeschreibung usw. Für die hier geplante Übung beschränken wir uns auf drei Textfelder, eines für den Autor, eines für den Titel, eines für das Sachgebiet und ein numerisches Feld für das Erscheinungsjahr.

ANLEGEN

Zum Anlegen oder Einrichten einer Datei dient der Befehl **anlegen**, den Sie gleich mit dem Namen für die neue Datei versehen. Archive stellt Ihnen automatisch die erforderlichen doppelten Anführungszeichen zur Verfügung. In den folgenden Zeilen geben Sie die Felder ein, aus denen der Datensatz bestehen soll. Das Dollarzeichen am Schluß kennzeichnet ein Feld als Textfeld; Feldbezeichnungen ohne Dollarzeichen sind numerische Felder. Wenn Sie alle gewünschten Felder definiert haben, beenden Sie den Vorgang mit dem Befehl **endeanlegen**. Die Sequenz zum Anlegen einer einfachen Bücher-Datei könnte beispielsweise so aussehen:

```
anlegen "Bücher"
Autor$
Titel$
Sachgebiet$
Jahr
endeanlegen
```

Beim Anlegen direkt von der Tastatur aus können Sie sich die Eingabe von **endeanlegen** auch sparen, indem Sie stattdessen auf einer leeren Zeile einfach **ENTER** drücken. Obligatorisch ist die Eingabe von **endeanlegen** jedoch beim Schreiben eines Archive-Programms.

DATENSÄTZE EINFÜGEN

Sobald eine neue Datei angelegt ist, kann sie gelesen und beschrieben werden. Im Moment ist sie aber natürlich leer. Zum Einfügen von Datensätzen verwenden Sie den Befehl

```
einfügen
```

worauf in der Maske folgendes angezeigt wird:

```
Logischer Name      : Haupt
Autor$              :
Titel$              :
Sachgebiet$         :
Jahr                :
```

Jetzt können Sie folgende Eintragungen vornehmen:

Wunderlich, J. [TAB]
Alle Geheimnisse gelüftet [TAB]
Einführung in die Computerei [TAB]
1984 [TAB]

worauf der neue Datensatz so aussieht:

```
Logischer Name      : Haupt
Autor$              : Wunderlich, J.
Titel$              : Alle Geheimnisse gelüftet
Sachgebiet$         : Einführung in die Computerei
Jahr                : 1984
```

Nun fügen Sie diesen Datensatz in die Datei namens "Bücher" ein, indem Sie die Taste **F5** drücken. Die Datenfelder werden geleert und sind für weitere Eintragungen verfügbar.

Zum Abschließen der einzelnen Feldinhalte und zum Übergang auf das nächste Feld können Sie statt **TAB** auch **ENTER** verwenden. **ENTER** nach Eingabe des letzten Datenfeldes bewirkt automatisch die Übernahme des Datensatzes in die Datei.

Wenn Sie mit dem Einfügen fertig sind oder aufhören wollen, drücken Sie **F4** und schließen die Datei mit **schließen** bzw. **verlassen**. Erst dann den Computer ausschalten.

KAPITEL 7
BILDSCHIRM-MASKEN

Beim **anzeigen** einer Datei erscheint diese in der Standard-Maske, d.h. mit dem von Archive standardmäßig vorgesehenen Bildschirmaufbau.

MASKEN AUFBAUEN

Es ist Ihnen jedoch freigestellt, eine eigene Maske zu definieren, wenn Ihnen für Ihre Art von Informationen eine andere Gliederung geeigneter erscheint. Geben Sie dazu den Befehl ein:

```
editor
```

Im Anzeigebereich wird die momentan eingerichtete Maske gezeigt, nämlich die von Archive automatisch vorgesehene. Unter Umständen ist der Anzeigebereich leer, falls sich im Arbeitsspeicher kein Bildschirm-Layout befindet.

Beim Betrachten der Maske fällt auf, daß die Werte der Felder nicht angegeben, sondern durch eine Reihe von Punkten markiert sind. Sie müssen sich die Maske als den Hintergrund vorstellen, auf dem eine Reihe von Variablenwerten in bestimmten, festgelegten Positionen eingetragen werden. Archive baut die Maske jeweils in zwei Phasen auf: zuerst den Hintergrundtext und dann die Werte der Variablen an den dafür vorgesehenen Positionen.

Sie befinden sich zunächst auf der *Hauptebene* des Befehls, von wo Sie zwischen drei Optionen wählen:

Hintergrundtext (Bildschirmmaske) eingeben
Editor mit **ESC** verlassen
Mit **F3** einen **Editor**-Befehl auswählen.

Wenn Sie eine eigene Bildschirmmaske entwerfen wollen, drücken Sie **F3** und dann **L** zum Leeren des Bildschirms. Mit **ENTER** diesen Schritt bestätigen; andernfalls mit einer beliebigen anderen Taste zur **Editor**-Hauptebene zurückkehren.

Bestimmen Sie die Farben für die Schrift und das Papier durch Drücken von **S** bzw. **P** und Farbauswahl mit Hilfe beliebiger Tasten. Mit **ENTER** kehren Sie zur Hauptebene zurück, von wo Sie den Hintergrundtext eingeben können, beispielsweise:

Hannes Meyers Bibliothek

Genausogut können Sie die Bezeichnung der Felder ändern oder ergänzen, z.B.

Bevölkerung (Mio.)

Der Cursor kann mit Hilfe der vier Pfeiltasten an jede beliebige Stelle in der Maske gebracht werden. Ihre Eingabe erscheint sofort an der betreffenden Stelle und ist von da an Teil des Bildschirmhintergrundes. Nicht zulässig ist eine Eingabe in den Bereichen, die für Variablen reserviert und mit Punkten markiert sind, es sei denn, Sie geben diese Plätze zur Eingabe frei. Darauf kommen wir später noch ausführlich zu sprechen. Beim Versuch, gepunktete Plätze zu überschreiben, gibt Archive im Arbeitsbereich den Namen der betreffenden Variablen an und akzeptiert keine Eingabe.

Die vier Bildschirm-Editorbefehle gestatten es Ihnen, eigene farbenfrohe Masken und Formate für die optimale Anzeige Ihrer Daten zu entwerfen. Wie der Bildschirm zu leeren ist, wissen Sie bereits. Zum Experimentieren mit den anderen drei **Editor**-Befehlen empfehlen wir Ihnen die Verwendung einer Reservekopie der Datei, um das Original intakt zu lassen.

BILDSCHIRM-EDITORBEFEHLE

Variable (V)

Nehmen wir an, Sie wollen den Wert der Variablen Land$ an einer bestimmten Position anzeigen. Plazieren Sie den Cursor an die betreffende Stelle, drücken Sie **F3** und danach **V**. Archive fordert die Eingabe des gewünschten Variablennamens. Antworten Sie mit

```
Land$
```

Die Eingabe führt nicht zum Anzeigen des Namens am Bildschirm, sie markiert lediglich die Stelle, wo der Wert erscheinen soll. Nach Drücken von **ENTER** fragt Archive nach der Breite, die reserviert werden soll. Durch Drücken einer beliebigen Taste (mit Ausnahme von **ENTER**) setzen Sie eine Reihe von Punkten zur Markierung der Länge. Überflüssiger Platz kann mit **CTRL** und der Linkspfeiltaste wieder gekürzt werden. Wenn Ihnen das Feld richtig zu sein scheint, kehren Sie mit **ENTER** zur **Editor**-Hauptebene zurück.

Beim Eintritt des Cursors in eine reservierte Zone reagiert Archive durch Anzeige des betreffenden Variablennamens im Arbeitsbereich.

Falls Sie für eine Variable einen Platz vorsehen, der einen bereits reservierten Bereich überschneidet, wird Ihnen Gelegenheit gegeben, den ursprünglich belegten Platz zu annullieren. Mit derselben Option können Sie einer neuen Variablen Platz zuweisen.

Schrift (S)

Zur Änderung der Schriftfarbe bewegen Sie den Cursor an die gewünschte Stelle und betätigen **F3**, gefolgt von **S**. Die aktuelle Farbe wird unten im Arbeitsbereich angezeigt und läßt sich durch Drücken einer beliebigen Taste ändern. Mit **ENTER** wird die gezeigte Farbe festgelegt. Danach eingegebener Text erscheint so lange in der gewählten Farbe, bis eine erneute Farbänderung vorgenommen wird.

Papier (P)

Analog verhält es sich mit der Änderung der Papierfarbe, nur daß hier nach **F3 P** gedrückt werden muß.

Es ist auch möglich, mitten in einer Zeile eine Farbänderung vorzunehmen. Markieren Sie den Anfang mit dem Cursor und wählen Sie Text- und Papierfarbe, fahren Sie dann mit dem Cursor ans Ende der farblich abgesetzten Textzone und stellen Sie wieder auf die ursprünglichen Farben um.

MASKE AKTIVIEREN

Sobald Sie mit Ihrem Maskenentwurf zufrieden sind, verlassen Sie den **Editor**. Von diesem Augenblick an ist der von Ihnen definierte Bildschirmaufbau aktiv, d.h. die Werte der Variablen in der Maske kommen automatisch zur Anzeige, wenn immer Archive einen Befehl oder ein Programm ausgeführt hat. Bei Eingabe des Befehls **folgesatz** z.B. springt Archive zum nächsten Datensatz in der aktuellen Datei und zeigt genau die Felder an, die Sie in Ihrer Maske definiert haben. Verwendung von **leeren** setzt den aktiven Bildschirmaufbau außer Kraft.

Ein nicht mehr aktiver (außer Kraft gesetzter) Bildschirmaufbau kann mit dem **schirm**-Befehl wieder aktiviert werden.

Dabei wird lediglich der Hintergrundtext angezeigt, jedoch keine aktuellen Variablenwerte.

MASKEN SICHERN UND LADEN

Die fertige Maske kann mit **bsichern** auf Microdrive-Kassette gesichert werden:

```
bsichern "Dateiname"
```

wobei "Dateiname" durch einen beliebigen Namen ersetzt werden kann. Die Maske wird genau in der am Bildschirm gezeigten Form gesichert.

Zum Laden einer Maske dient der Befehl:

```
bladen "Dateiname"
```

der die Maske nicht nur in den Arbeitsspeicher lädt, sondern auch gleich am Bildschirm anzeigt und aktiviert.

Aus einem Archive-Programm heraus werden die angezeigten Werte nicht automatisch ersetzt, wenn sich der aktuelle Datensatz ändert. Angenommen, Sie wollen sich alle Datensätze aus der aktuellen Datei der Reihe nach anzeigen lassen und verwenden dazu das einzeilige Programm:

```
anf: setzen x=0: solange x<anzahl():folgesatz:setzen x=x+1:
    endesolange
```

Die Befehle **solange** und **endesolange** bewirken die wiederholte Ausführung des dazwischenliegenden Programmteils, solange die Bedingung nach **solange** zutrifft (wahr ist). Jeder **solange-Befehl** muß mit einem **endesolange** gepaart sein.

Dieses Kurzprogramm erzielt nicht die gewünschte Wirkung, da Archive den Inhalt des Bildschirms immer erst am Ende des Programms aktualisiert.

DER BEFEHL BANZEIGE

Hingegen ist es möglich, eine Anzeige der Variablenwerte in einer aktiven Maske aus einem Programm heraus zu veranlassen. Dafür gibt es den Befehl **banzeige**, der in dieser Programmzeile verwendet wird und zur beabsichtigten Anzeige aller Datensätze führt:

```
anf:setzen x=0:solange x<anzahl():banzeige:folgesatz:
    setzen x=x+1:endesolange
```

Ohne eine aktivierte Maske bleibt **banzeige** ohne Wirkung.

DER BEFEHL ANZEIGEN

Denken Sie daran, daß der **anzeigen**-Befehl immer den Standard-Bildschirmaufbau verwendet. Er ersetzt jede benutzerdefinierte Maske durch die Archive-spezifische. Um zu verhindern, daß Ihre selbst definierte Maske unwiederbringlich verlorengeht, müssen Sie sie unbedingt mit **bsichern** abspeichern, bevor Sie **anzeigen** verwenden.

KAPITEL 8
PROZEDUREN

Für die Übungen in diesem Kapitel geben Sie zunächst den **neu**-Befehl ein, um den Arbeitsspeicher zu leeren, und dann **sichten** "Staaten" zum Eröffnen der Demonstrationsdatei. Es wird davon ausgegangen, daß diese sich im Microdrive 2 befindet.

Die Befehle und Funktionen von Archive bilden zusammen eine mächtige Programmiersprache, mit der Sie eigene Programme für die Manipulation Ihrer Dateien schreiben können. Sie werden staunen, wie einfach es ist, Archive-Programme zu erstellen.

Ein Archive-Programm besteht aus einem oder mehreren separaten Teilen, den sog. *Prozeduren*. Eine Prozedur ist nichts anderes als ein Unterprogramm mit einem eigenen Namen, mit dem sie aufgerufen werden kann – genau wie die Prozeduren in SuperBASIC. In Archive betreibt man Prozeduren einfach durch Eingabe des betreffenden Namens über die Tastatur. Eine Prozedur ist im Prinzip nichts anderes als ein neuer Befehl, welcher dem eingebauten Archive-internen Befehlsvorrat hinzugefügt wird.

Prozeduren dürfen nicht länger als 255 Zeilen sein, und jede Zeile darf maximal 160 Zeichen enthalten.

PROZEDUREN ERSTELLEN

Zum Schreiben und Überarbeiten von Prozeduren verwenden Sie den *Programmeditor*, der alle Funktionen zum Ändern, Löschen und Hinzufügen von Text enthält.

Für eine ausführliche Beschreibung des Programmeditors verweisen wir auf Kapitel 9. An dieser Stelle befassen wir uns nur mit einigen Funktionen, die wir zum Erstellen von ein paar kurzen Prozeduren brauchen. Wir gehen davon aus, daß der Arbeitsspeicher noch keine Prozeduren enthält.

Geben Sie ein:

```
editieren
```

zum Eintritt in den Programmeditor. Die Anzeige im Steuerbereich ändert sich und enthält die Anweisung, den Namen der Prozedur einzugeben. Der Eintritt in den Programmeditor gestattet Ihnen das Erstellen einer neuen Prozedur, es sei denn, es ist bereits eine Prozedur im Speicher vorhanden.

Auch hier müssen wir uns als erstes überlegen, was genau die neue Prozedur überhaupt bewerkstelligen soll. Beginnen wir mit einer ganz unkomplizierten Prozedur, die uns etwas Tipparbeit spart. Statt des ausgeschriebenen Befehls **anzeigen** wollen wir von nun an nur noch A eingeben.

Schreiben Sie einfach

```
a
```

In der Rubrik links im Anzeigebereich erscheint nun der Prozedurname (a) und rechts davon eine Auflistung unserer Prozedur, die noch etwas fragmentarisch aussieht. **Proz** und **EndeProz** markieren den Anfang und das Ende und wurden von Archive automatisch eingefügt.

Der *Hauptteil* der Prozedur, welcher die Anweisungen enthält, muß jetzt dazwischen eingeschoben werden.

Wie Sie aus den Angaben im Steuerbereich sehen, erwartet Archive jetzt die Eingabe von Zeilen in die Prozedur. In unserem Beispiel geht es um den **anzeigen**-Befehl. Schreiben Sie also:

```
anzeigen
```

worauf Archive die Zeile sofort vor EndeProz einfügt, so daß unsere Prozedur jetzt dieses Aussehen hat:

```
a Proz a
   anzeigen
   EndeProz
```

In der selben Weise könnten Sie weitere Befehlszeilen eingeben. Die zuletzt eingegebene wird jeweils farblich vom Rest abgesetzt.

Für unser Beispiel reicht dies jedoch aus, die Prozedur ist vollständig definiert, und Sie verlassen **editieren** durch zweimaliges Drücken von **ESC**.

Zum Betreiben der Prozedur brauchen Sie nichts weiter zu tun, als ihren Namen (a) einzugeben, gefolgt von **ENTER**. Damit sparen Sie sich das Ausschreiben des Befehls **anzeigen.**

PROZEDUREN AUFLISTEN UND DRUCKEN

Bei jedem Aufruf von **editieren** präsentiert Archive eine Liste der Namen aller definierten Prozeduren im Arbeitsspeicher.

Für eine Auflistung einer bestimmten Prozedur innerhalb des **editieren**-Befehls bewegen Sie sich mit **TAB** in der Liste nach unten und mit **SHIFT** plus **TAB** nach oben, bis der gesuchte Prozedurname hervorgehoben wird. Es erfolgt dann eine automatische Auflistung dieser Prozedur in der rechten Rubrik. Bei langen Prozeduren, die nicht als ganze in den Anzeigebereich passen, wird nur der Anfang angezeigt, und Sie können sich dann mit den Auf– und Abwärtspfeilen in der Prozedur auf– und abbewegen. Wenn Sie fertig sind, verlassen Sie **editieren** mit **ESC**.

Für einen Papierausdruck der Prozedurauflistung verwenden Sie den Befehl

```
listen
```

welcher alle momentan im Arbeitsspeicher befindlichen Prozeduren über den Drucker ausgibt.

VORSICHT: Vor Verwendung dieses Befehls sicherstellen, daß ein funktionsfähiger Drucker angeschlossen ist, weil sonst das Programm hängenbleibt.

PROZEDUREN SICHERN UND LADEN

Wenn Sie Ihre Prozeduren für den späteren Wiedergebrauch aufbewahren wollen, sichern Sie sie mit dem **sichern**–Befehl. Damit speichern Sie alle definierten Prozeduren in einer einzigen benannten Datei auf Microdrive-Kassette. Zum Abspeichern der soeben definierten Prozedur (a) können Sie eine Datei mit dem Namen "Proz1" vorsehen, so daß der Befehl lautet:

```
sichern "Proz1"
```

Bei Bedarf holen Sie sich die Prozedur wieder in den Arbeitsspeicher, indem Sie den Befehl

```
laden "Proz1"
```

eingeben. Dabei muß daran gedacht werden, daß der **laden**-Befehl alle zu dem Zeitpunkt im Arbeitsspeicher befindlichen Prozeduren löscht und mit der neu geladenen überschreibt. Dieses Problem läßt sich mit dem **mischen**-Befehl vermeiden. Wenn Sie also stattdessen

```
mischen "Proz1"
```

verwenden, wird "Proz1" in den Arbeitsspeicher geladen, ohne die bereits dort existierenden Prozeduren zu vernichten. Allerdings würde die neue Prozedur eine gleichnamige, bereits bestehende Prozedur überschreiben.

PROZEDUREN ÜBERPRÜFEN

Eine Methode, Arbeit und Zeit zu sparen, ist das Ersetzen langer Befehlsnamen durch einen einzigen Buchstaben, wie wir es oben mit **anzeigen** demonstriert haben. Eine weitere Möglichkeit besteht darin, mit Hilfe einer längeren Prozedur mehrere Befehle durch einen einzelnen Tastendruck zu ersetzen. Versuchen Sie, mit **editieren** die folgende Prozedur zu definieren, die Ihnen gestattet, eine beliebige Datenbank zu eröffnen und zu prüfen, vorausgesetzt natürlich, die Datei ist nicht bereits geladen.

Falls Sie sich noch im **editieren**-Befehl befinden, drücken Sie **F3**, gefolgt von **N** für "Neue Prozedur".

Lassen Sie sich nicht aus der Ruhe bringen, wenn Ihnen beim Abschreiben der Prozedur Fehler passieren. Wie Sie sie korrigieren können, erfahren Sie im nächsten Kapitel.

```
Proz ANSICHT
leeren
eingabe "Welche Datei? ";Datei$
sichten Datei$
anzeigen
setzen Taste$="t"
solange Taste$<>"v"
  banzeige
  setzen Taste$=klein(tippen())
  wenn Taste$="a":anf:endewenn
  wenn Taste$="e":ende:endewenn
  wenn Taste$="f":folgesatz:endewenn
  wenn Taste$="z":zurück:endewenn
  endesolange
schließen
EndeProz
```

Daran denken, **editieren** durch zweimaliges Drücken von **ESC** zu verlassen und die Prozedur anschließend zu sichern.

Sie können die Prozedur jetzt durch Eingabe von

```
ansicht
```

aufrufen. Als erstes leert sie den Bildschirm. Dann fordert sie die Eingabe des Dateinamens, beispielsweise "Staaten". Falls diese Datei bereits geladen ist, resultiert die Eingabe in einer Fehlermeldung. Geben Sie **neu** ein und laden und betreiben Sie die Prozedur noch einmal. Die Prozedur eröffnet die genannte Datenbank im Nur-zum-Lesen-Modus und zeigt den ersten Datensatz in der Maske an. Sie wartet auf eine Tasteneingabe und reagiert auf A, E, F, Z bzw. V. Die ersten vier bewirken die Anzeige des entsprechenden Satzes (anf, ende, folgesatz, zurück), während V (verlassen) die Datei schließt und die Prozedur beendet.

So – das wäre unser erstes Programm respektabler Länge und vielleicht keine schlechte Gelegenheit für ein paar Tips zum Schreiben von Programmen. Sicher ist Ihnen aufgefallen, daß die Programmzeilen auf dem Bildschirm verschieden weit eingerückt sind. Das macht die Struktur übersichtlich und erleichtert das Lesen. Sie brauchen sich jedoch beim Eingeben nicht mit Leerzeichen abzumühen: die Einrückungen erfolgen automatisch beim Schreiben, Auflisten oder Ausdrucken der Prozedur.

Der Hauptteil der Prozedur (Tastendruck abwarten und entsprechende Aktion einleiten) ist zwischen den beiden zusammengehörigen Befehlen **solange** und **endesolange** eingeschlossen. Man spricht von einer *Programmschleife*, die wiederholt ausgeführt und erst abgebrochen wird, wenn die Bedingung direkt nach **solange** falsch ist. In unserem Fall wird die Bedingung in dem Moment falsch (unwahr), wo die Taste "V" angeschlagen wird.

Eine ähnliche Art von Schleife bilden die beiden Befehle **wenn** und **endewenn**, die auch stets als Paar auftreten und so lange ausgeführt werden wie die Bedingung wahr ist. Es sind zwei separate Befehle, die nicht in der gleichen Zeile stehen müssen. Die erste **wenn**-Anweisung könnte genausogut wie folgt formuliert werden:

```
wenn Taste$="a"
  anf
  endewenn
```

Die Schleife zwischen **wenn** und **endwenn** kann mehrere Anweisungen umfassen, die alle unter der Voraussetzung ausgeführt werden, daß die Bedingung direkt nach **wenn** wahr ist. In unserer ANSICHT-Prozedur sind die Anweisungen so kurz, daß jeweils die ganze Schleife auf einer Zeile Platz findet, wobei die einzelnen Anweisungen mit Doppelpunkten zu trennen sind.

Wie Sie sehen, enthält die Prozedur auch einen **banzeige**-Befehl in der Hauptschleife, der sicherstellt, daß jeder neue Datensatz am Bildschirm angezeigt wird. Sie erinnern sich, daß **anzeigen** zwar immer die Standardmaske aktiviert, jedoch keine automatische Veränderung des Anzeigebereichs bewirkt. Ohne den Befehl **banzeige** würden keine Datensätze angezeigt, bis die Taste "V" zum Verlassen der Prozedur gedrückt wird. Sie würden folglich nur das Ergebnis des zuletzt ausgeführten Tastendrucks zu Gesicht bekommen.

KAPITEL 9
DER PROGRAMM-EDITOR

Dieses Kapitel beschreibt den Programmeditor. Es enthält auch ein paar einfache Übungsbeispiele, doch werden Sie sich bestimmt sehr schnell mit dem **Editor** anfreunden, wenn Sie ihn selbst bei Ihrer Arbeit benutzen.

Wir würden vorschlagen, daß Sie nach der Lektüre dieses Kapitels entweder selbst einige einfache Programme verfassen oder die Prozeduren aus dem vorangehenden Kapitel modifizieren. Und wenn Sie lieber gleich mit längeren Beispielen üben möchten, gibt es in den restlichen Kapiteln genügend Programme zum Eintippen.

EDITIEREN

Der Befehl **editieren** bringt Sie auf die *Hauptebene* des Programmeditors.

Als Beispiel wollen wir eine Prozedur erstellen und einige Anweisungen hinzufügen. Aus der Hauptebene drücken Sie **F3** und **N** zum Einrichten einer neuen Prozedur. Beantworten Sie die Frage nach dem Namen mit **Test**.

Zum sofortigen Verlassen des Programmeditors und Rückkehr auf die Hauptebene zweimal **ESC** drücken. Dann erneut **editieren** aufrufen. Wenn keine anderen Prozeduren geladen sind, zeigt die Bildschirmmaske jetzt:

```
Test   Proz Test
       EndeProz
```

Wenn sich die früher definierten Prozeduren noch im Arbeitsspeicher befinden, erkennen Sie in der Rubrik links die verschiedenen Namen. Zum Übergang auf Einfügemodus **F4** drücken, worauf die Zeile **"Proz..."** die Farbe wechelt.

Geben Sie diese beiden Programmzeilen ein:

```
zeigen "Das ist ein Test" [ENTER]
zeigen "mit zwei Anweisungen" [ENTER] [ENTER]
```

Das zweimalige Drücken von **ENTER** bringt Sie heraus aus dem Einfügemodus. Die Prozedur hat jetzt folgendes Aussehen:

```
Test   Proz Test
          Zeigen "Das ist ein Test"
          Zeigen "mit zwei Anweisungen"
          EndeProz
```

wobei die Programmzeile mit der zweiten Anweisung in einer anderen Farbe erscheint.

Bevor Sie **ENTER** drücken, können Sie Korrekturen und Änderungen mit Hilfe des Zeileneditors vornehmen. Sobald Sie jedoch die Programmzeile mit **ENTER** abgeschlossen haben, ist die Anweisung fest in die Prozedur eingefügt. Soll sie nochmals überarbeitet werden, holen Sie sie mit **F5** heraus, machen die erforderliche Korrektur und drücken wieder **ENTER**. Diese korrigierte Version ersetzt dann die vorherige Zeile.

Die Abschlußanweisung **EndeProz** kann nicht geändert werden und ebensowenig der Ausdruck **Proz** zu Anfang der ersten Zeile. Hingegen können Sie den Namen der Prozedur ohne weiteres ändern, indem Sie den alten Namen mit dem Zeileneditor überschreiben. Die Liste der Prozedurnamen in der Rubrik links im Anzeigebereich wird automatisch alphabetisch geordnet.

EDITIERBEFEHLE

Es gibt vier Editierbefehle, die im mittleren Feld des Steuerbereichs angezeigt werden, wenn Sie eine Prozedur erstellen. Angewählt werden Sie mit **F3** plus dem betreffenden Anfangsbuchstaben.

Neue Prozedur (N)

Sie geben den Namen der neuen Prozedur ein, die Sie schreiben wollen. Bei Vorhandensein einer Prozedur gleichen Namens wird diese durch die neu definierte überschrieben.

Mit Drücken von **ENTER** nach Eingabe des Namens machen Sie die neue Prozedur zur aktuellen. Sie wird damit rechts von dem Trennbalken aufgelistet. Im Augenblick besteht sie jedoch lediglich aus den Anfangs- und Endanweisungen **Proz** und **EndeProz**.

Löschen Prozedur (L)

Dieser Befehl löscht die aktuelle Prozedur aus Ihrem Programm. Wählen Sie die betreffende Prozedur mit **TAB** bzw. **SHIFT** plus **TAB**, um sie zur aktuellen Prozedur zu machen. Rufen Sie dann mit **F3** und **L** den Löschbefehl.

Der Löschvorgang wird erst ausgeführt, wenn Sie ihn mit **ENTER** bestätigen. Vorher haben Sie noch Zeit, es sich anders zu überlegen und die Prozedur mit dem **editieren**-Befehl zu ändern, statt sie zu löschen.

Vorsicht beim Umgang mit löschen: eine einmal gelöschte Prozedur ist unwiderruflich verloren und muß neu eingetippt werden, falls sie versehentlich gelöscht wurde.

Wegnehmen (Zeilen) (W)

Dieser Befehl löscht eine oder mehrere Anweisungszeilen aus der aktuellen Prozedur. Auf diese Weise entfernte Zeilen können bei Bedarf an einer anderen Stelle eingefügt werden, und zwar sowohl in der gleichen wie auch in einer anderen Prozedur. Dafür gibt es den **einsetzen** (**E**) Befehl.

Vor Aufruf von **wegnehmen** müssen Sie anhand der Cursorsteuertasten entweder die erste oder die letzte Zeile des Programmteils, der entfernt werden soll, zur aktuellen Zeile machen. Anschließend rufen Sie den Befehl mit **F3** und **W** auf.

Eingabe von **ENTER** entfernt die aktuelle Zeile aus der Prozedur. Soll mehr als eine einzelne Zeile entfernt werden, bewegen Sie jetzt den Cursor auf das andere Ende des Zeilenblocks, der dabei farbig hinterlegt wird. Wenn Sie dann **ENTER** drücken, extrahiert Archive den markierten Zeilenblock und plaziert ihn in einen reservierten Speicherbereich, einen sog. Puffer.

Einsetzen (Zeilen) (E)

Dieser Befehl setzt eine Zeile bzw. einen Zeilenblock aus der vorangehenden **wegnehmen**-Operation in die aktuelle Prozedur ein, und zwar direkt unter der aktuellen Zeile. Auf Wunsch kann der Text an einer anderen Position oder sogar in einer anderen Prozedur eingesetzt werden.

Vor Aufrufen des **E**-Befehls ist gegebenenfalls mit **SHIFT** und **TAB** die Prozedur auszuwählen, in die der Text eingefügt werden soll. Ferner markieren Sie mit den Pfeiltasten die Zeile, unter welcher der Einschub erfolgen soll.

Archive fügt die Zeilen bzw. den Zeilenblock sofort direkt aus dem Puffer unter die aktuelle Zeile ein. Der Puffer ist damit leer und kann nicht noch einmal an einer anderen Stelle eingesetzt werden.

KAPITEL 10
PROGRAMMIEREN MIT ARCHIVE

Dieses Kapitel beschreibt die Entwicklung eines realistischen Beispiels mit allen erforderlichen Arbeitsschritten, die ausführlich erklärt werden.

Stellen Sie sich vor, Sie seien Sekretär eines Vereins, der einen Jahresbeitrag erhebt und eine Zeitung herausgibt. Jedem Mitglied, das seinen Beitrag entrichtet hat, wird ein Exemplar der Zeitung zugeschickt, und vor dem Fälligkeitsdatum erhalten alle eine Zahlungsaufforderung.

Das im folgenden dargestellte Beispiel gestattet die Erstellung einer Versandliste und bei Bedarf das Ausdrucken von Klebeadressen, welche zusätzlich einen Vermerk über das Fälligkeitsdatum enthalten. Es geht von der Annahme aus, daß jährlich sechs Zeitungsausgaben versandt werden und daß der Mitgliederbeitrag bei Erhalt der letzten Ausgabe fällig wird. Das Beispiel ließe sich ohne weiteres auf jede Situation anpassen, wo in regelmäßigen Abständen eine Art Rundschreiben oder Serienbrief an eine Reihe von Adressaten einer Versandliste geschickt werden muß.

VERSANDLISTE

Wir wollen in diesem Beispiel die bereits bekannten Funktionen optimal ausnützen und auch einige neue einführen. Wenn Sie irgendwo nicht ganz klarkommen mit Befehlen oder Funktionen, die Ihnen nicht vertraut sind oder deren Wirkungsweise Sie nicht verstehen, empfehlen wir Ihnen, die betreffende Kurzbeschreibung im Übersichtsteil nachzulesen oder die **Hilfe-Datei** mit **F1** zu konsultieren. Zum Verändern der Datensätze machen wir ausschließlich Gebrauch von den beiden Befehlen **einfügen** und **ändern**, während wir zum Ausdrucken der Adreß-Etiketten auf spezielle, selbst geschriebene Routinen angewiesen sind.

Wir haben es mit folgenden Aufgaben zu tun:

Hinzufügen neuer Datensätze in die Datenbank
Löschen von Datensätzen
Ändern von Datensätzen
Buchen eingegangener Mitgliederbeiträge
Herstellen der Etiketten
Verlassen des Programms.

Für die Bewältigung jeder dieser Aufgaben werden wir eine eigene Prozedur schreiben und am Schluß alle diese Prozeduren durch eine andere so verknüpfen, daß jeweils die gewünschte Aufgabe bearbeitet werden kann.

Für unsere Zwecke ist es ganz klar, was für Felder die Datensätze enthalten müssen. Selbstverständlich Felder für Name und Anschrift und dazu noch ein Feld für die Anzahl der an die betreffende Person verschickten Zeitungsausgaben.

Wir können also sofort die erforderliche Datei anlegen:

```
anlegen "post"
   Titel$
   Vorname$
   Name$
   Straße$
   PLZ_Ort$
   Land$
   Ausgaben
   Endeanlegen
```

Wir haben hier drei separate Felder für Titel (z.B. Frau, Herr, Dr.), und für den Vornamen und Namen eingerichtet, obwohl bestimmt auch ein einziges Feld ausgereicht hätte.

Für die Anschrift wurden ebenfalls drei Felder vorgesehen, eines für die Straße, eines für Postleitzahl und Ort und eines für das Land. Die Felder brauchen nicht immer in dieser Weise belegt zu werden und können einfach allgemein als drei für die Anschrift reservierte Felder gelten. Das sollte in den meisten Fällen mehr als ausreichen.

Außerdem wurde ein numerisches Feld Ausgaben für die Anzahl der noch zu verschickenden Zeitungsausgaben vorgesehen.

Damit hätten wir unsere Datei angelegt, und wir werden von nun an die Prozeduren, die wir schreiben, daran testen. Es ist in jedem Fall eine gute Idee, selbst geschriebene Prozeduren nach Möglichkeit immer sofort zu testen und nicht bis zum Schluß damit zu warten, wenn alles sehr viel verzwickter ist und verschiedene Dinge gleichzeitig

schiefgehen können. Versuchen Sie stets, in der Testphase alles so einfach und durchschaubar wie nur möglich zu halten, und nehmen Sie eine neue Prozedur erst in Angriff, wenn Sie sicher sind, daß die vorangehende einwandfrei funktioniert. Auf diese Weise erleben Sie keine bösen Überraschungen, sondern haben im Gegenteil ein Programm, das nach Fertigstellen der letzten Prozedur auf Anhieb klappt.

Einfügen

Zum Einfügen eines neuen Datensatzes brauchen wir uns keine eigene Prozedur einfallen zu lassen. Archive hat ja bereits den eingebauten sehr bequemem Befehl **einfügen**. Vergessen Sie nicht, in einer Prozedur **banzeige** zu verwenden, um die Anzeige des Satzinhalts zu veranlassen. Mit Hilfe von **einfügen** können Sie jetzt gleich einige Datensätze eintragen, damit Sie danach die anderen Prozeduren an einer echten Datei testen können.

Löschen

Irgendwann kommt der Zeitpunkt, wo Sie jene Mitglieder aus der Kartei streichen wollen, die ihre Mitgliedschaft nicht erneuert haben. Zu diesem Zweck wollen wir eine separate Prozedur namens **raus** definieren, mit der Sie die ganze Datei auf Personen durchsehen können, die ihren Beitrag schuldig geblieben sind, um zu entscheiden, welche gestrichen werden sollen.

```
Proz raus
  anm *****nicht-zahlende Mitglieder ausstreichen*****
  leeren
  anzeigen
  wählen Ausgaben =0
  alle
    banzeige
    zeigen in 10,0; "Löschen (J/N)? ";
    setzen OK$ =klein(taste())
    zeigen OK$
    wenn OK$ ="j"
      löschen
      zeigen "Gelöscht"
      endewenn
    endealle
  rücksetzen
  EndeProz
```

Da ein einmal gelöschter Datensatz unwiderruflich gelöscht ist, wird vorsichtshalber der gesamte Inhalt des Datensatzes angezeigt und die Löschprozedur erst durchgeführt, wenn Sie die Bestätigung geben. Wir machen Gebrauch von der Funktion **taste()**, welche einen Tastendruck abwartet und als Wert das entsprechende Zeichen liefert. Die Funktion **klein()** wandelt in den entsprechenden Kleinbuchstaben um, so daß es unerheblich ist, welche Schreibweise Sie verwenden.

Wollen Sie es wagen, diese Prozedur an Ihrer Datei auszuprobieren? Das geht selbstverständlich nur, wenn Sie dort bereits ein paar Sätze eingetragen haben. Verlassen Sie den Programmeditor durch ein– bzw. zweimaliges Drücken von **ESC** und speichern Sie die Prozedur "raus" auf eine Datei mit dem Namen "versand":

```
sichern "versand"
```

Damit ist die Prozedur namens **raus** abgespeichert und ist jederzeit abrufbereit, nach dem "versand" geladen wurde.

Jede der nachfolgenden Prozeduren ist in der gleichen Weise in der Datei "versand" abzuspeichern.

Zahlungen

Meist wird es so sein, daß Sie eine ganze Serie von Beitragszahlungen aus einer Liste mit Namen und Anschriften eintragen wollen. Dazu brauchen Sie die einzelnen Datensätze. Um die Suche zu beschleunigen, schreiben wir eine eigene Prozedur, **holen**, zum Auffinden eines bestimmten Datensatzes, die wir dann in eine Prozedur namens **zahlung** einbauen.

Die **holen**-Prozedur fragt nach einer Zeichenkette (n$) und ermittelt dann den ersten Datensatz in der Datei, der diese Suchbedingung erfüllt. Drücken von **ENTER** ohne Texteingabe setzt n$ auf eine Nullkette, was eine Suche verhindert. Folglich dient es auch als Signal, daß alle Eintragungen gemacht wurden.

Aus der **editieren**-Ebene **F3** gefolgt von **N** drücken, um die Prozedur **holen** zu definieren:

```
Proz holen
  anm ***** Datensatz herausholen *****
  leeren
  setzen OK$ ="n"
  eingabe "WER? "; n$
  wenn n$ <>""
    finden n$
    solange OK$ <>"j" und gefunden()
    zeigen titel$ ; " "; vorname$(1); " "; name$
    zeigen straße$
    setzen OK$ =klein(taste())
    leeren
    wenn OK$ <>"j"
      weiter
      endewenn
    endesolange
  wenn nicht gefunden()
    zeigen in 12,0; n$ ; "Suche erfolglos"
    endewenn
  endewenn
EndeProz
```

Die Suche arbeitet mit dem **finden**-Befehl, der alle Textfelder nach der eingegebenen Zeichenkette absucht. Datensätze können folglich nach dem Namen oder nach der Anschrift identifiziert werden. Natürlich ist nicht immer der erste Datensatz bereits der richtige, und aus diesem Grund muß die Suche weitergeführt werden können. Genau das bewerkstelligt die **solange .. endesolange** Schleife, welche den Namen und die erste Zeile der Adresse anzeigt und anfragt, ob dies der gewünschte Datensatz sei. Die Suche wird so lange fortgesetzt, bis Sie mit J reagieren. Das ist die Abbruchbedingung für die Schleife, d.h. sie hält an, wenn J gedrückt wird, bzw. wenn der Suchbegriff in den restlichen Datensätzen nicht auftaucht. Beachten Sie, daß die Funktion **gefunden()** "wahr" (einen Wert ungleich Null) ausgibt, wenn die Suche erfolgreich verläuft.

Da der Anfangswert von OK$ von einer vorangegangenen erfolgreich abgeschlossenen Suche normalerweise auf "j" steht, muß er zu Beginn der Prozedur, vor dem Eintritt in die Schleife, geändert werden. Dies garantiert, daß die Schleife mindestens einmal durchlaufen wird.

Und jetzt zur Prozedur **zahlung**:

```
Proz zahlung
  anm *****Bezahlte Mitgliedsbeiträge*****
  leeren
  setzen n$ = "x"
  solange n$ <>""
    holen
    wenn OK$ ="j"
    setzen ausgaben =ausgaben + 6
    ersetzen
    endewenn
  endesolange
EndeProz
```

Die hierin enthaltene Schleife wird so lange durchlaufen, bis n$ eine Nullkette ist. Auf diese Weise können Sie jeweils gleich mehrere Zahlungen auf einmal eintragen, ohne für jede einzelne die Prozedur aufrufen zu müssen. Wenn Sie mit Ihren Eintragungen fertig sind, beantworten Sie die Frage "Wer?" einfach durch Drücken von **ENTER**, ohne jegliche Eingabe.

Wenn der Wert von OK$ nach dem Aufruf von **holen** "j" ist, wird die Zahlung aufgezeichnet und als gültig für sechs weitere Zeitungsausgaben gekennzeichnet.

Auch hier muß der Anfangswert von n$ wieder auf einen geeigneten Wert gesetzt werden (beliebiger Wert, nur keine Nullkette), um sicherzustellen, daß die Prozedur nicht durch eine etwaige vorangehende Operation beeinträchtigt wird.

Änderungen

Eine Prozedur zum Ändern des Inhalts eines Datensatzes ist nicht schwer zu schreiben. Da es auch hier wieder darum geht, einen bestimmten Satz auszuwählen und zu ändern, kann die Struktur im großen und ganzen von der Prozedur **zahlung** übernommen werden:

```
Proz modifikation
   anm *****Datensatz modifizieren*****
   setzen n$ ="x"
   leeren
   solange n$ <>""
      holen
      wenn OK$ ="j"
         ändern
         leeren
         endewenn
      endesolange
EndeProz
```

PARAMETER

An dieser Stelle wollen wir eine kurze Unterbrechung in der Entwicklung unseres Programms einlegen und uns mit etwas Neuem beschäftigen: der Verwendung von *Parametern* im Zusammenhang mit Prozeduren. Ein Parameter kann dazu dienen, einen Wert an eine Prozedur zu übergeben. Um die Sache etwas konkreter zu machen, werden wir die Idee gleich an ein paar Beispielen veranschaulichen. Diese brauchen Sie nicht etwa in der "Versand"-Datei zu speichern, sondern können sie löschen, bevor Sie mit unserem Demonstrationsprogramm und dem Ausdrucken von Etiketten weiterfahren.

Hier ein ganz einfaches Beispiel, um Ihnen eine Vorstellung zu vermitteln:

```
Proz Probe; a
   zeigen 5*a
   EndeProz
```

Damit wird eine Prozedur namens **Probe** definiert, die einen Parameter (a) verlangt. Beachten Sie, daß der Parameter vom Prozedurnamen durch ein Semikolon abgetrennt ist. Jedesmal, wenn Sie diese Prozedur aufrufen, müssen Sie ihr auch einen Wert für diesen Parameter mitgeben. Die Eingabe hat dann eine Form von der Art:

```
Probe; 3
```

und liefert das Resultat 15. Der eingegebene Wert (3) ist an die Prozedur als Wert der Variablen a übergeben und von dieser ausgewertet worden.

Eine Prozedur kann eine beliebige Anzahl von Parametern enthalten, die durch Kommas voneinander abzugrenzen sind. Hier ein Beispiel einer Prozedur mit drei Parametern:

```
Proz Experiment; a,b,c
zeigen a * b * c
EndeProz
```

die z.B. so aufgerufen wird:

```
Experiment; 3,4,5
```

Die beim Aufruf mitgegebenen Werte brauchen keine Konstanten zu sein; genausogut können Variablen verwendet werden:

```
setzen x = 2
setzen y = 5
setzen z = 7
Experiment; x,y,z
```

Wichtig ist noch, daß die Namen der Variablen nicht identisch sein müssen mit den innerhalb der Prozedur vorkommenden Namen. Man unterscheidet zwischen den *formalen Parametern* (z.B. a,b,c) in der Definition einer Prozedur und den *aktuellen Parametern*, d.h. den effektiv an die Prozedur übergebenen Werten.

Auch Resultate von mathematischen Ausdrücken können an Prozeduren übergeben werden, z.B.

```
Experiment; x*2,z/y,(z-y)*x
```

Und damit noch nicht genug: außer numerischen Variablen sind auch Zeichenketten als Parameter zulässig, vorausgesetzt, Sie kennzeichnen die formalen Parameter in der Prozedur mit dem üblichen Dollarzeichen. Ein Beispiel zum Ausprobieren:

```
Proz Versuch; a$
zeigen a$
EndeProz

setzen t$ = "Nachricht"
Versuch; t$
```

Die einzige Bedingung bei Verwendung von Parametern in Prozeduren ist die Übereinstimmung zwischen den formalen Parametern in der Definition und den effektiv übergebenen Parametern beim Prozeduraufruf hinsichtlich Anzahl und Typ (numerisch, Text).

Adreß-Etiketten

Das kurze Intermezzo über Parameter haben wir mit Absicht hier eingeschoben. Mit Hilfe von Parametern ist es möglich, eine elegante Prozedur zum Schreiben der Etiketten zu formulieren. Für Testzwecke ist es am besten, die Prozedur erst so zu definieren, daß die Etiketten am Bildschirm angezeigt und erst später für Druckerausgabe umgewandelt werden können. Wir gehen davon aus, daß die Etiketten acht Zeilen lang werden. Falls dies für Ihren Drucker oder Ihre Klebeadressen nicht geeignet ist, müssen Sie die entsprechende Zahl in der Prozedur ändern. Vergessen Sie nicht, die nun folgenden Prozeduren wieder in der "Versand"-Datei zu sichern.

Zunächst schreiben wir eine Prozedur, die eine einzelne Zeile ausgibt, deren Inhalt anhand eines Parameters übergeben wird:

```
Proz Zeile; x$
zeigen x$
EndeProz
```

Sie kann nunmehr in der folgenden Prozedur für die Ausgabe der Adresse integriert werden:

```
Proz Etikett
   anm *****Etiketten drucken*****
   wenn ausgaben
      wenn ausgaben =1
         Zeile; "MAHNUNG - Beitrag jetzt fällig"
         sonst
         Zeile; ""
         endewenn
   Zeile; ""
   Zeile; titel$ +" "+vorname$ (1)+". "+name$
   Zeile; straße$
   Zeile; PLZ_Ort$
   Zeile; Land$
   Zeile; ""
   setzen ausgaben =ausgaben - 1
   ersetzen
   endewenn
EndeProz
```

Wie Sie sehen, enthält die Prozedur auch einen Vermerk über den Fälligkeitstermin, der bei all jenen Mitgliedern angebracht wird, die gerade ihre letzte Ausgabe erhalten. Beim Drucken jedes Etiketts wird der Zählerstand der ausgelieferten Zeitungen um 1 verringert. Erreicht der Zählerstand 0, wird kein Etikett gedruckt.

Inzwischen können Sie sich sicher schon vorstellen, wie nützlich Parameter sind. Diese Prozedur würde erheblich länger, wenn wir ohne Parameter auskommen müßten. Beachten Sie auch, wie einfach es ist, Titel, Vorname und Name für die Adresse in eine einzige Zeile zu setzen (mit Hilfe des Verknüpfungs-Operators +).

Vielleicht fragen Sie sich, weshalb wir uns die Mühe gemacht haben, die kurze Prozedur **Zeile** zu definieren, wo wir doch genausogut einfach **zeigen**-Anweisungen in **Etikett** hätten verwenden können. Auch dafür gibt es einen guten Grund. So wie die Prozedur jetzt formuliert ist, erfolgt die Ausgabe der Etiketten am Bildschirm. Um sie an den Drucker zu leiten, brauchen wir lediglich eine Anweisung in **Zeile** zu modifizieren – und nicht etwa alle **zeigen**-Anweisungen in **Etikett**. Die einzige Änderung, die wir vornehmen müssen, ist:

```
Proz Zeile; x$
drucken x$
EndeProz
```

Zu guter letzt schreiben wir noch die Prozedur zum Ausdrucken der Etiketten:

```
Proz senden
   leeren
   alle
      etikett
      endealle
   EndeProz
```

Programm verlassen

Bleibt noch die Option zum Verlassen des Programms nach Beendigung der Arbeit. Diese Prozedur ist denkbar einfach. Sie muß nur die Datei ordnungsgemäß abschließen und dann die Steuerung wieder an die Tastatur zurückgeben. Wir haben zusätzlich noch eine Kurzmeldung eingefügt, die anzeigt, daß das Programm zu Ende ist.

```
Proz Adieu
schließen
leeren
zeigen in 6,25; "Adieu"
stopp
EndeProz
```

FEHLER

Vermutlich wird Ihnen irgendwann beim Arbeiten mit diesem Programm ein Fehler passieren, etwa ein unbeabsichtigtes Drücken der **ESC**-Taste oder die versehentliche Eingabe von Text in ein numerisches Feld. Archive entdeckt diese Art von Fehlern und quittiert sie im allgemeinen mit einer Fehlernachricht und Übergabe der Steuerung vom Programm an die Tastatur.

Mit dem **fehler**-Befehl können Prozeduren markiert werden, die in einer Fehlersituation speziell zu behandeln sind. Bei Auftreten eines Fehlers innerhalb der auf diese Weise gekennzeichneten Prozedur bzw. in einer durch Sie aufgerufenen, erfolgt ein sofortiger, frühzeitiger Rücksprung.

Die allgemein übliche Fehlerbehandlung ist bei solchen speziell markierten Prozeduren *ausgeschaltet*, und es bleibt Ihnen überlassen, wie Sie auf den Fehler reagieren wollen. Die Fehlernummer ermitteln Sie anhand der Funktion **fehlernr()**, die mehrmals zum Ablesen verwendet werden kann, da ihr Wert erst beim nächsten Gebrauch des **fehler** – Befehls wieder auf Null gesetzt wird. Wenn das Programm seit Anfang bzw. seit der letzten Ausführung von **fehler** ohne Fehler verlaufen ist, hat **fehlernr()** den Wert Null.

Der **fehler**-Befehl verursacht vielleicht zu Anfang einiges Kopfzerbrechen, gibt Ihnen jedoch große Flexibilität und Entscheidungsfreiheit bei der Behandlung von Fehlern. Das folgende Beispiel veranschaulicht eine typische Anwendung, eine Prozedur, welche die Benutzereingabe von numerischen Werten narrensicher macht:

```
Proz Test1
   eingabe x
   EndeProz

Proz Test2
   setzen n =1
   solange n
      fehler test1
      setzen n =fehlernr()
      wenn n
         zeigen "Fehler Nr. ";n ;"entdeckt. Bitte versuchen
            Sie's nochmal"
         endewenn
      endesolange
   EndeProz
```

Test1 wartet darauf, daß Sie der Variablen x einen Wert zuweisen. Test2 behandelt beliebige bei Ausführung der Eingabeprozedur auftretende Fehler. Taucht innerhalb von Test1 ein Fehler auf, wird die Prozedur frühzeitig abgebrochen und die Fehlernummer gesetzt. Diese Zahl wird durch **fehlernr()** eingelesen und bewirkt, wenn sie ungleich Null ist, die Anzeige der vorgesehenen Fehlernachricht. Den Text dazu können Sie sich frei

ausdenken. Diese Anweisungen bilden eine **solange..endesolange** Schleife, die bei jedem auftretenden Fehler ausgeführt wird. Die Fehlernummer wird durch **fehler** wieder gelöscht und zum nächsten Versuch freigegeben. Test2 kann nicht verlassen werden, bevor eine gültige Eingabe gemacht wurde.

Diese Beispielprozedur meldet die Nummer des entdeckten Fehlers. In den meisten Fällen ist jedoch die Art des Fehlers für Sie von geringem Interesse; die eigentliche Funktion von **fehlernr()** ist die Unterscheidung zwischen Situationen ohne Fehler und solchen, bei denen ein Fehler entdeckt wird. Eine Zusammenstellung der Fehlernummern, zusammen mit den möglichen Ursachen, finden Sie in der Übersicht am Schluß dieses Abschnitts.

Nun sind wir in der Lage, eine Prozedur zu formulieren, die es Ihnen gestattet, durch einen einzigen Tastendruck eine aus den sechs Optionen auszuwählen. Sie ist so einfach im Aufbau, daß sich eine Erklärung erübrigt.

```
Proz Wahl
   anm ***** Eine Option auswählen*****
   leeren
   zeigen
   zeigen "Einfügen Senden Zahlung Modifikation Raus Verlassen";
   zeigen "? ";
   setzen Wahl$ = klein(taste())
   zeigen Wahl$
   wenn Wahl$ ="e": einfügen : endewenn
   wenn Wahl$ ="s": senden : endewenn
   wenn Wahl$ ="z": zahlung : endewenn
   wenn Wahl$ ="m": modifikation : endewenn
   wenn Wahl$ ="r": raus : endewenn
   wenn Wahl$ ="v": adieu : endewenn
EndeProz
```

Was noch zu tun bleibt, um unser Programm zu vervollständigen, ist eine Prozedur zum Eröffnen der Datei und zum Aufrufen von **Wahl**. Damit Sie die Optionen jedesmal neu vorgelegt bekommen, wenn eine beendet wurde, muß die Prozedur in eine Schleife eingebettet werden.

Die **solange..endesolange** Schleife in der zu diesem Zweck vorgesehenen Prozedur läuft immer und immer weiter. Es handelt sich um eine sog. Endlosschleife, die nur dann abbricht, wenn der Ausdruck direkt nach **solange** Null ist. Da der Ausdruck jedoch immer 1 bleibt, wird die Abbruchbedingung niemals zutreffen. Die Schleife wird unaufhörlich durchlaufen. Der einzige Ausweg ist durch Wahl der V-(Verlassen)-Option. Der **stopp**-Befehl in **Adieu** übergibt die Steuerung sofort wieder an die Tastatur.

```
Proz Start
   anm *****Start-Prozedur*****
   leeren
   eröffnen "post"
   solange 1
      fehler wahl
      setzen n =fehlernr()
      wenn n
         zeigen in 12,0; "Fehler - beliebige Taste
         zum Weitermachen"
         setzen m$ =taste()
      endewenn
   endesolange
EndeProz
```

Diese Schleife enthält eine Reihe von Anweisungen, welche alle auftretenden Fehler in einer ähnlichen Weise behandeln wie im vorangehenden Abschnitt beschrieben. Wenn Sie einen Fehler machen, hält das Programm mit der Verarbeitung inne, bis Sie eine Taste drücken. Auf diese Weise haben Sie Gelegenheit, den gemachten Fehler aufzuspüren.

PROGRAMME AUSFÜHREN

Die Hauptprozedur in unserem Programm ist "Start". Sie erlaubt Ihnen, zum Betreiben des Programms den **ausführen**-Befehl zu verwenden.

Sichern Sie auch diese letzte Prozedur in der "Versand"-Datei. Bevor Sie das Programm betreiben können, müssen die Prozeduren in den Arbeitsspeicher geholt werden. Dann wird die Hauptprozedur ausgeführt, welche alle anderen der Reihe nach aufruft. Sie können so vorgehen, daß Sie zuerst den **laden**-Befehl verwenden und dann den Namen der Hauptprozedur eingeben:

```
laden "versand"
start
```

Der **ausführen**-Befehl bewältigt diese beiden Schritte automatisch, d.h. er lädt das genannte Programm und führt die Prozedur namens "Start" aus, wenn eine solche vorhanden ist. Sie brauchen also lediglich einzugeben:

```
ausführen "versand"
```

Die restlichen beiden Abschnitte in diesem Kapitel enthalten eine Reihe von Mehrzweck-Prozeduren, die Ihnen vielleicht für Ihre eigenen Anwendungen auch gute Dienste leisten.

LOKALE VARIABLEN

Die meisten Variablen innerhalb von Prozeduren sind sog. *globale*, d.h. allgemeingültige Variablen, die im ganzen Programm erkannt werden. Sie können in einer beliebigen Prozedur verwendet oder modifiziert werden – nicht nur in der ersten, in der sie definiert wurden.

Im Gegensatz dazu sind die als formale Parameter verwendeten Variablen *lokale Variablen*, deren Geltungsbereich örtlich beschränkt ist, nämlich auf die Prozedur, in der sie auftreten.

Das folgende Beispiel soll den Unterschied verdeutlichen. Leeren Sie zunächst den Arbeitsspeicher durch Eingabe von **neu**. Wir wollen jetzt eine Prozedur erstellen, die zwei lokale Variablen, a und b$, verwendet und außerdem zwei normale, globale Variablen namens u und v$.

```
Proz Demo; a,b$
zeigen a;b$
setzen u=3
setzen v$="Text"
zeigen u;v$
EndeProz
```

Wenn wir "Demo" mit

```
demo;5,"Wörter"
```

aufrufen, werden alle vier Werte ausgegeben, was bedeutet, daß alle vier Variablen innerhalb von "Demo" identifiziert werden konnten.

Die Eingabe von

```
zeigen u;v$
```

beweist, daß diese beiden Variablen auch außerhalb der "Demo"-Prozedur erkannt werden. Hingegen wird durch die Eingabe von

```
zeigen a;b$
```

ein Fehler provoziert, weil die beiden lokalen Variablen a und b$ außerhalb der Prozedur nicht bekannt sind.

Alle formalen Parameter sind lokale Variablen, doch können auch andere Variablen als solche deklariert werden, beispielsweise:

```
Proz Max
zeigen "in Max"
zeigen p; q; r
EndeProz

Proz Moritz
lokal q,r
setzen p = 2
setzen q = 3
setzen r = 4
zeigen "in Moritz"
zeigen p; q; r
Max
EndeProz
```

Bei Verwendung von "Moritz" mit dem Aufruf:

```
Moritz
```

zeigt sich, daß alle Werte von p, q und r in "Moritz" erkannt und daher ausgegeben werden, während "Max" mit den Variablen q und r nichts anfangen kann, weil diese moritz-spezifisch definiert (lokal) sind.

Die Werte von lokalen Variablen sind nur in der Prozedur definiert, in der sie auch erklärt werden. Ihr Geltungsbereich erstreckt sich nicht auf Prozeduren, die von dieser ersten aufgerufen werden. Globale Variablen hingegen sind Allgemeingut und werden im ganzen Programm erkannt.

Vielleicht wundern Sie sich, weshalb man überhaupt lokale Variablen einführt. Um ihre Nützlichkeit an einem Beispiel zu erläutern, nehmen wir an, Sie schreiben ein Programm, welches aus mehreren Prozeduren besteht, die Sie entweder selbst früher für ein anderes Programm definiert haben oder die Sie von jemand anderem übernehmen. Dabei kann es ohne weiteres vorkommen, daß mehrere dieser Prozeduren Variablen verwenden, die ganz verschiedene Aufgaben erfüllen, jedoch zufällig denselben Namen haben. Wenn es sich dabei um globale Variablen handelt, beeinflussen und stören sie sich gegenseitig, was zu völlig ungewollten Effekten führt. Es gäbe keinen anderen Weg, als jede Prozedur auf ihre Variablennamen zu untersuchen und diese gegebenenfalls zu ändern. Dieses Problem taucht nicht auf, wenn die Variablen als lokal deklariert wurden, weil dann jede unabhängig von allen anderen funktioniert und modifiziert werden kann.

Es macht auch nichts aus, wenn eine Prozedur eine andere aufruft, die eine Variable mit demselben Namen enthält – solange mindestens eine von beiden als lokal deklariert worden ist. In der Prozedur **Wahl** z.B. wurde eine lokale Variable namens auswahl$ eingeführt. Dabei erübrigte es sich, nachzuprüfen, ob irgend eine der vielen durch **Wahl** aufgerufenen Prozeduren unter Umständen eine gleichnamige Variable benutzt, weil eine Beeinflussung lokaler Variablen völlig ausgeschlossen ist.

EINGABE ANFORDERN

Beim interaktiven Arbeiten mit dem Computer kommt es häufig vor, daß der Benutzer aufgefordert wird, eine Eingabe vorzunehmen, d.h. dem Computer eine Information über die Tastatur einzugeben. Es lohnt sich, für die Ausgabe solcher Aufforderungen eine allgemeine Prozedur zu schreiben. Eine einfache Methode zur Ausgabe einer Nachricht ist, diese in Form eines Parameters an die Prozedur zu übergeben.

```
Proz Meldung; m$
zeigen m$ + ": ";
setzen x$ =klein(taste())
zeigen x$
EndeProz
```

Die Meldung, die ausgegeben werden soll, wird in der lokalen Variablen m$ übergeben. Die Funktion **taste()** wartet einen Tastendruck ab und liefert das entsprechende Zeichen, das durch die Funktion **klein()** auf Kleinbuchstaben umgewandelt wird, so daß es unwichtig ist, ob ein Groß- oder ein Kleinbuchstabe eingegeben wird. Schließlich wird das Resultat der Variablen x$ zugewiesen. Dabei handelt es sich um eine globale Variable, welche die Eingabe auch anderen Prozeduren innerhalb des Programms verfügbar macht.

PAUSE

Eine weitere praktische Prozedur ist "Pause". Sie verwendet "Meldung" zur Anzeige einer Nachricht und wartet dann, bis der Benutzer eine Taste drückt. Im allgemeinen ist es von geringem Interesse, welche Taste angeschlagen wird, und so wird der ursprüngliche Wert von x$ einer lokalen Variablen, y$, zugewiesen.

```
Proz Pause
   anm ***** Auf Tastenanschlag warten *****
   lokal y$
   setzen y$ =x$
   zeigen
   meldung; "Beliebige Zeichentaste zum Weitermachen drücken"
   setzen x$ = y$
EndeProz
```

DATENEINGABE

Text

Die Übernahme von Text als Eingabe über die Tastatur ist ganz und gar unproblematisch. Jede beliebige Aneinanderreihung von Zeichen ist eine gültige Textkette, auch wenn sie inhaltlich gesehen keinen Sinn ergibt. Im allgemeinen sind bei der Anforderung von Texteingabe keine besonderen Vorkehrungen notwendig. Eine Zeile wie die nachstehend aufgeführte reicht aus, um den Benutzer nach dem Namen zu fragen. Selbstverständlich muß die Variable durch Anhängen eines "$" als Textvariable gekennzeichnet sein.

```
eingabe "Wie ist Ihr Name, bitte: ";name$
```

Beachten Sie das Leerzeichen nach dem Doppelpunkt, welches zwar nicht obligatorisch ist, jedoch den optischen Eindruck der Meldung am Bildschirm erheblich verbessert.

Eine einzige Eingabeanweisung kann gleich mehrere Elemente enthalten. Schreiben Sie die Anforderung und die Variablennamen und trennen Sie die einzelnen Elemente jeweils durch ein Semikolon z.B.:

```
eingabe "Ihr Vorname? ";vname$;"Ihr Familienname? ";fname$;
```

Diese letzte Anweisungszeile endet mit einem Semikolon. Damit wird verhindert, daß der Cursor nach erfolgter Eingabe auf die nächste Zeile weiterspringt.

Zahlen

Bei Verwendung des **eingabe** – Befehls zur Aufforderung von Texteingabe akzeptiert der Computer jede beliebige alphanumerische Zeichenkette (Buchstaben und Zahlen). Viel pingeliger ist er bei numerischen Variablen. Jede Eingabe, die etwas anderes als eine erlaubte Zahl ist, wird mit einer Fehlermeldung und dem Ausstieg aus dem Programm quittiert. Allerdings ist es möglich, sich gegen solche Unfälle zu schützen, indem man dem Programm Instruktionen zur Handhabung derartiger Eingabefehler gibt.

Hier kommt der **fehler**-Befehl wie gerufen, den wir ja bereits kennengelernt haben. Die folgende Prozedur akzeptiert jede gültige Zahl innerhalb des angegebenen Bereichs und gibt darüberhinaus eine vorgegebene Nachricht aus.

```
Proz holenum; m$,min,max
   anm ***** Nummer im Zahlenbereich holen *****
   lokal falsch
   setzen falsch=1
   solange falsch
      zeigen m$; "? ";
      fehler Liesnum
      setzen falsch=fehlernr()
      wenn nicht falsch
         wenn num<min oder num>max
         setzen falsch=1
         zeigen "Zugelassener Bereich ist ";min;" bis ";max
         endewenn
      endewenn
      wenn falsch
         zeigen "Bitte nochmals versuchen"
         endewenn
      endesolange
EndeProz
```

Da dem **fehler**-Befehl der Name einer Prozedur folgen muß, definieren wir **Liesnum** so, daß es der Variablen **Num** einen Wert zuweist.

```
Proz Liesnum
eingabe Num
EndeProz
```

Angenommen, Sie möchten eine Prozedur, die prüft, daß die Zahl im Bereich zwischen 1 und 10 liegt, dann bietet sich die Verwendung von **holenum** wie folgt an:

```
Proz test
holenum; "Numerischer Wert",1,10
EndeProz
```

KAPITEL 11

ARBEITEN MIT MEHREREN DATEIEN

LOGISCHE DATEINAMEN

Dieses Kapitel eröffnet weitere Möglichkeiten der Programmierung mit Archive, die sich durch die gleichzeitige Manipulation mehrerer Dateien ergeben. Sobald Sie mit mehr als nur einer eröffneten Datei arbeiten, muß für jede Operation die gewünschte Datei genannt werden. Zu diesem Zweck weisen Sie jeder Datei beim Eröffnen bzw. beim Anlegen einen eindeutigen *logischen Namen* zu, der bei allen künftigen Operationen zum Ansprechen dieser Datei verwendet wird.

Archive gibt unaufgefordert den logischen Dateinamen "Haupt" aus, wenn eine Einzeldatei eröffnet wird. Die Bezeichnung *Logischer Dateiname* steht im Gegensatz zum *physischen Dateinamen*, den Sie einer Datei vor dem Abspeichern zuweisen.

Programme benutzen stets die logischen Dateinamen. Sie können also ein Programm schreiben, welches verschiedene Dateien bearbeitet.

Die logischen Dateinamen sind unerläßlich, wenn eine Operation an mehreren Dateien ausgeführt werden soll, weil eine zweite Datei nur geöffnet werden kann, wenn sowohl ihr logischer wie auch ihr physischer Name genannt werden. Zu beachten ist dabei, daß der logische Name beim Schließen der Datei nicht mit abgespeichert wird, sondern jedesmal beim Eröffnen neu vergeben werden kann.

Es ist denkbar, daß zwei oder mehr Dateien Felder mit dem gleichen Namen enthalten. In solchen Fällen kann das betreffende Feld durch Angabe des logischen Namens eindeutig identifiziert werden. Wenn beispielsweise das Feld Land$ in zwei Dateien vorkommt, deren logische Namen "Haupt" und "B" lauten, dann werden sie durch die ausführlichen Bezeichnungen "Haupt.Land$" und "B.Land$" auseinandergehalten.

SÄTZE EINER DATEI ÄNDERN

Unser erstes Beispiel illustriert das Einfügen, Löschen und Umbenennen von Feldern in bestehenden Dateien.

Angenommen, wir wollen die "Staaten"-Datei so verändern, daß wir daraus eine neue Datei zusammenstellen können, die nur europäische Länder enthält. Dabei wird das Feld "Kontinent" überflüssig und kann weggelassen werden. Außerdem soll die abgekürzte Form "Bev" als "Bevölkerung" ausgeschrieben werden.

Am einfachsten ist es, eine zweite Datei mit den gewünschten Feldern anzulegen und dann die erforderlichen Informationen aus der ursprünglichen Datei zu kopieren. Nennen wir unsere neue Datei "Europa". Den Rest der Arbeit überlassen wir getrost der folgenden Prozedur:

```
Proz Start
  anm ***** Europa-Datei anlegen *****
  anlegen "Europa" logisch "e"
    Land$
    Hauptstadt$
    Sprachen$
    Währung$
    Bevölkerung
    BSP
    Fläche
    EndeAnlegen
  sichten "Staaten" logisch "s"
  wählen Kontinent$="EUROPA"
  alle "s"
    zeigen in 0,0;s.Land$;tab 35
    setzen e.Land$=s.Land$
    setzen e.Hauptstadt$=s.Hauptstadt$
    setzen e.Sprachen$=s.Sprachen$
    setzen e.Währung$=s.Währung$
    setzen e.Bevölkerung=s.Bev
```

```
        setzen e.BSP=s.BSP
        setzen e.Fläche=s.Fläche
        erweitern "e"
        endealle
      schließen "e"
      schließen "s"
      zeigen
      zeigen "FERTIG"
    EndeProz
```

DIE AKTUELLE DATEI

Dieses Beispiel zeigt, daß es ohne weiteres möglich ist, einem Feld in zwei verschiedenen Dateien denselben Namen zu geben. Eine eindeutige Identifikation kann jederzeit durch Hinzufügen des logischen Namens erfolgen. Fehlt ein solcher, geht Archive davon aus, daß die *aktuelle Datei* gemeint ist. Wenn mehrere Dateien nacheinander eröffnet werden, so ist immer die zuletzt eröffnete Datei auch die aktuelle Datei. In unserem Beispiel ist die aktuelle Datei "Staaten" (logischer Name "s"), so daß wir jeweils auf das "s" vor dem Feldnamen auch hätten verzichten können.

In allen Fällen, wo die Angabe des logischen Namens fakultativ ist, führt Archive die angeforderte Operation an der aktuellen Datei aus. Es empfiehlt sich jedoch, den logischen Namen stets ausdrücklich mitanzugeben, um möglichen Konfusionen aus dem Weg zu gehen.

Sie können zu jedem Zeitpunkt bestimmen, welche Datei die aktuelle sein soll. Dafür gibt es den **aktiv**-Befehl. Mit der Anweisung

```
    aktiv "e"
```

im obigen Programm würde "Europa" zur aktuellen Datei, und zwar so lange, bis Sie den Befehl ausdrücklich ändern bzw. bis zur Eröffnung einer anderen Datei.

LAGERHALTUNG

Zur Abwechslung wollen wir uns ein etwas komplexeres Beispiel vornehmen. Das nun folgende Beispiel für eine Lagerhaltung ist kein perfektes Programm, sondern soll eine Hilfe sein, wie Sie eine eigene komplexe Applikation erstellen können. Eine Lager-haltung müßte folgende Aufgaben bewältigen können:

Informationen zu bestimmten Lagerposten herausholen
Den gegenwärtigen Lagerbestand aller Lagerposten ermitteln
Verkäufe registrieren und Lagerbestand entsprechend ändern
Bestellungen zum Auffüllen des Lagers vornehmen
Lieferungen (Wareneingänge) registrieren.

Zu diesem Zweck brauchen Sie ganz sicher eine Datei, welche alle Angaben zu den einzelnen Lagerposten enthält, und dazu vielleicht eine zweite mit Informationen über die Lieferanten. Sie müssen in der Lage sein, von einer Datei auf die andere zuzugreifen, etwa um herauszufinden, welche Lieferanten für einen bestimmten Lagerposten in Frage kommen oder was für Posten beim gleichen Lieferanten bestellt werden können.

Im Hinblick auf größtmögliche Einfachheit werden wir bei diesem Beispiel, anders als bei den vorangehenden, nicht eine menügesteuerte Anwendung vorschlagen, sondern stattdessen eine Reihe von separaten Befehlen definieren, die genau wie die eingebauten Archive-Befehle durch Eingabe ihres Namens aufgerufen werden.

Da die Prozeduren weitgehend von der Dateistruktur abhängen, lohnt es sich, erst einige Gedanken darauf zu verwenden.

Die Lager-Datei

Die Lager-Datei muß alle Einzelheiten über den Lagerbestand jedes Postens enthalten. Die folgende Aufstellung erläutert die verschiedenen Felder:

Feldname	Inhalt	Beispiel
`Lager_Nr$`	Firmeninterne Lager-Nr.	`A101`
`Bezeichnung$`	Kurzbeschreibung des Postens	`Besen, Plastik`
`Menge`	Anzahl auf Lager	`500`
`V_Preis`	Verkaufspreis, netto	`22,40`
`Min_Bestand`	Neubestellen, wenn Bestand unter diesen Wert fällt	`200`
`Bestellung`	Bestellmenge	`400`

Legen wir also unsere Datei an:

```
anlegen "lager" logisch "la"
lager_nr$
bezeichnung$
menge
V_Preis
min_bestand
bestellung
endeanlegen
```

Die Lieferanten-Datei

In dieser Datei wollen wir die Namen, Anschriften und Telefonnummern der Firmen aufführen, von denen Sie Ihre Waren beziehen. Unter Umständen ist es von Vorteil, auch den Namen der Kontaktperson anzugeben, mit der Sie normalerweise verhandeln. Zur optimalen Verwaltung aller Informationen geben wir auch jeder Lieferfirma einen Code. Die folgenden Felder sollen eingerichtet werden:

Feldname	Inhalt	Beispiel
Firma$	Name des Lieferanten	Besenwunder AG
Straße$	Erste Zeile der Adresse	Mühlgasse 38
PLZ_Ort$	Postleitzahl und Ort	7000 Stuttgart
Land$	Land	BRD
Kontakt$	Name, Verhandlungspartner	Kurt Kübler
Tel$	Telefonnummer	0707-334455
Code$	Ihr Lieferanten-Code	B

Die Datei kann jetzt angelegt werden:

```
anlegen "liefer" logisch "lf"
Firma$
Straße$
PLZ_Ort$
Land$
Kontakt$
Tel$
Code$
endeanlegen
```

Die Bestell-Datei

Diese Datei bildet die Verbindung zwischen den beiden vorangehenden. Sie besteht aus den folgenden Feldern:

Feldname	Inhalt	Beispiel
Lager_Nr$	Firmeninterne Lager-Nr.	A101
Code$	Ihr Lieferanten-Code	B
Artikel$	Bestell-Nr.	123-456
Preis	Einkaufspreis	16,80
Lieferzeit	Lieferzeit (Tage)	17

Jeder Datensatz in dieser Datei verknüpft einen Satz aus der Lager-Datei mit einem aus der Lieferanten-Datei. Aus obigem Beispiel geht hervor, daß Sie von der Firma Besenwunder (Lieferanten-Code "B") Plastikbesen (Lager-Nr. "A101") beziehen können. Außerdem können Sie sich über Preis, Liefertermine und die Bestellnummer informieren – alles Angaben, die Sie bei der Bestellung brauchen.

Diese Datei kann auch Fälle bewältigen, wo ein Lieferant als Bezugsquelle für mehrere Warenposten in Frage kommt (identische Werte im Feld Code$, jedoch unterschiedliche bei Lager__Nr$) und solche, wo ein Posten von mehr als einem Lieferanten nachbestellt werden kann (identische Lager__Nr$, jedoch verschiedene Werte unter Code$).

Anlegen der Datei:

```
anlegen "bestell" logisch "be"
Lager_Nr$
Code$
Artikel$
Preis
Lieferzeit
endeanlegen
```

Anfragen

Nachdem wir diese Dateien angelegt haben, brauchen wir noch eine Prozedur, welche die Informationen sinnvoll verwaltet. Am häufigsten tritt wohl die Situation auf, wo auf Anfrage eines Kunden Informationen über einen bestimmten Warenposten ausfindig gemacht werden müssen. Der Suchvorgang soll möglichst wenig Zeit in Anspruch nehmen, und das Suchkriterium zum Auffinden des gewünschten Datensatzes ist entweder die Artikel-Nummer oder die Bezeichnung. Folglich benutzen wir den **finden**-Befehl in einer Weise, die Ihnen gestattet, die Suche durch Eingabe einer zulässigen Zeichenkette zu starten.

Die Prozedur muß in der Lage sein, nachzufragen, ob der gefundene Datensatz auch tatsächlich der gesuchte ist. Es lohnt sich, dafür eine eigene Prozedur zu definieren, die dann bei Bedarf auch in anderen Situationen eingesetzt werden kann.

```
Proz bestätigen
zeigen : zeigen "Bestätigt (J/N)";
setzen Ja=klein(taste())="j"
leeren
EndeProz
```

Bei Drücken der J-Taste erhält die Variable **Ja** den Wert 1; andernfalls ist er Null. Beachten Sie die Verwendung des "=" Zeichens – einmal als Zuweisungsoperator (bei setzen) und einmal als mathematisches Gleichheitszeichen in einer Bedingung.

```
Proz anfrage2
  anm ***** Anfrage betr. Lagerposten *****
  zeigen
  eingabe "Lagerposten? ";name$
  aktiv "la"
  finden name$
  setzen ja=0
  solange gefunden() und nicht ja
    anzeigen
    banzeige
    bestätigen
    wenn nicht ja
      weiter
      endewenn
    endesolange
  wenn nicht gefunden()
    zeigen
    zeigen name$; " nicht vorhanden"
    endewenn
EndeProz
```

Diese Prozedur dient lediglich zum Auffinden des gesuchten Datensatzes. Praktischer zum Befragen der Datei ist "anfrage"

```
Proz anfrage
  anfrage2
  freibild
EndeProz
```

die eine zusätzliche Prozedur, "freibild" enthält, welche einen Tastendruck abwartet, den Bildschirm leert und dann eine Liste der verfügbaren Befehle anzeigt. Diese Prozedur schreiben wir erst am Schluß, wenn wir alle Befehle definiert haben, die in der Liste erscheinen sollen.

Vergessen Sie nicht, von Zeit zu Zeit aus dem **editieren**-Befehl auszusteigen und die Prozeduren, die Sie eingetippt haben, abzuspeichern.

Lager Wir können auch ohne viel Mühe eine Prozedur definieren, die uns einen Überblick über den Lagerbestand gibt.

```
Proz lager
  leeren
  zeigen Tab 2; "POSTEN"; Tab 11; "CODE";
  zeigen Tab 20; "MENGE"; Tab 31; "PREIS";
  zeigen Tab 40; "LAGERWERT";
  zeigen
  setzen total=0
  aktiv "la"
  alle
    setzen k=länge(bezeichnung$)
    wenn k>10
      zeigen bezeichnung$(bis 10);
      sonst
      zeigen bezeichnung$;
      endewenn
    zeigen Tab 11;la.lager_nr$;
    zeigen Tab 20;Menge;
    zeigen Tab 31;"DM ";V_Preis; Tab 40;"DM ";V_Preis*Menge
    setzen total=total+V_Preis*Menge
    endealle
  zeigen
  zeigen "Lagerwert insgesamt = DM ";total
  freibild
EndeProz
```

Verkaufsstatistik Um einen Verkauf zu registrieren, wird die Anzahl der verkauften Warenposten aus dem entsprechenden Datensatz in der Lager–Datei subtrahiert. Auch hier ist es eine gute Idee, eine Bestätigung einzubauen, um sicherzustellen, daß wir den richtigen Datensatz vorliegen haben, und ferner, daß der Lagerbestand ausreicht, um die Verkaufsmenge zu liefern.

```
Proz Bestand
  anm ***** Vorhandener Warenbestand *****
  anfrage2
  zeigen
  eingabe "Anzahl? "; num
  leeren
  zeigen num;" * ";la.Lager_Nr$;" (";la.Bezeichnung$;")"
EndeProz

Proz Verkauf
  anm ***** Vorgang verarbeiten *****
  Bestand
  wenn num<=la.Menge
  zeigen "Auftragswert:DM ";num*la.V_Preis
  bestätigen
  wenn ja
    setzen la.Menge=la.Menge-num
    ersetzen
    banzeige: anm *** aktualisierten Datensatz vorzeigen ***
    endewenn
  sonst
  zeigen "Warenbestand nicht ausreichend"
  endewenn
  freibild
EndeProz
```

Hereinkommende Lieferungen Unsere nächste Prozedur registriert hereinkommende Lieferungen. Auch hier erfolgt die Modifikation und Aktualisierung des Datensatzes erst, nachdem Sie die Bestätigung geben.

```
Proz Eingang
  Bestand
  bestätigen
  zeigen
  wenn ja
    zeigen "Akzeptiert"
```

```
        setzen la.Menge=la.Menge+num
        setzen la.bestellung=la.bestellung-num
        ersetzen
        banzeige
        sonst
        zeigen ''Lieferung nicht aufgezeichnet''
        endewenn
      freibild
      EndeProz
```

Neubestellung

Die Prozeduren, die wir bis jetzt definiert haben, beziehen sich alle nur auf die Lager-Datei. Sobald es darum geht, Waren nachzubestellen, sind wir jedoch auf die Lieferanten-Datei angewiesen, aus der wir Name und Anschrift der Firmen und deren Preise entnehmen können.

Gehen wir davon aus, daß wir den richtigen Datensatz in der Lager-Datei mit "Anfrage" gefunden haben. Aus der Bestell-Datei suchen wir nun jene Datensätze heraus, die die richtige Lager-Nr. enthalten und damit auch unsere Codes für alle in Frage kommenden Lieferanten. Da die Datensätze außerdem auch die Preise und die Liefertermine ausweisen, können wir aufgrund dieser Informationen entscheiden, ob dem niedrigeren Preis oder der kürzeren Lieferfrist der Vorzug gegeben werden soll.

Zum schnellen Auffinden des erforderlichen Lieferanten-Satzes verwenden wir den **orten**-Befehl. Die Verwendung von **Bestellen** setzt eine (nach Code$) geordnete Lieferantendatei voraus.

```
Proz Bestellen
  anfrage2
  aktiv "be"
  wählen la.Lager_Nr$=be.Lager_Nr$
  zeigen
  aktiv "la"
  eingabe "Anzahl ? ";la.bestellung
  ersetzen
  aktiv "be"
  zeigen "hurtig oder billig (h/b)";
  wenn klein(taste())="h"
    hurtig
    sonst : billig
    endewenn
  setzen ycode$=code$
  rücksetzen
  aktiv "lf"
  orten verg$
  formular
  zeigen
  zeigen "Voraussichtlicher Liefertermin: ";lie;" Tage"
  freibild
  EndeProz
```

Die Prozedur **billig** ermittelt den Lieferanten mit dem günstigsten Preis, während **hurtig** jenen mit der kürzesten Lieferfrist ausfindig macht.

```
Proz Billig
  anm ***** den billigsten Lieferanten ausfindig machen *****
  aktiv "be"
  setzen Pr=Preis
  setzen verg$=code$
  setzen lie=lieferzeit
  alle
    wenn Preis<Pr
      setzen Pr=Preis
      setzen verg$=code$
      setzen lie=lieferzeit
      endewenn
    endealle
  EndeProz
```

```
Proz Hurtig
  anm ***** den schnellsten Lieferanten finden *****
  aktiv "be"
  setzen lie=lieferzeit
  setzen verg$=code$
  setzen Pr=Preis
  alle
    wenn lieferzeit<lie
      setzen lie=lieferzeit
      setzen verg$=code$
      setzen Pr=Preis
      endewenn
    endealle
  EndeProz
```

Die Prozedur **Formular** erstellt das eigentliche Bestellformular, welches Sie Ihren eigenen Bedürfnissen anpassen können. Wir verwenden hier eine einfache Version zur Anzeige der Bestell-Angaben am Bildschirm.

```
Proz Formular
  leeren
  zeigen
  zeigen lf.firma$
  zeigen lf.straße$
  zeigen lf.PLZ_Ort$
  zeigen lf.land$
  zeigen
  zeigen "Lieferung erbeten von "; la.bestellung;
  zeigen " * ";la.bezeichnung$;
  zeigen " zu DM ";Pr;" pro Stk."
  zeigen
  zeigen "Gesamtwert: DM ";la.bestellung*Pr
  EndeProz
```

Und zu guter letzt noch der Befehl zum Schließen aller Dateien, wenn die Arbeit zu Ende ist.

```
Proz adieu
  bestätigen
  wenn ja
    leeren
    zeigen : zeigen "Adieu"
    schließen "la"
    schließen "lf"
    schließen "be"
    endewenn
  EndeProz
```

So, und jetzt schreiben wir eine kurze Prozedur zum Betreiben des Anwendungsprogramms. Sie muß alle drei Dateien mit ihren logischen Namen eröffnen, die Bildschirmmaske leeren und eine Liste der verfügbaren Befehle, die wir definiert haben, anzeigen. Es sei noch darauf hingewiesen, daß in den meisten Fällen die Lager-Datei die einzige ist, an der Änderungen vorgenommen werden, da die Informationen in den anderen beiden unverändert bleiben. Aus diesem Grund reicht es aus, die Lieferanten- Datei ausschließlich zum Lesen zu eröffnen (mit **sichten**). Außerdem sortiert die Prozedur die Lieferantendatei, damit wir eine Firma nach dem Code ermitteln können.

```
Proz Start
  leeren
  zeigen in 5,5; "DEMONSTRATION LAGERHALTUNG"
  zeigen
  eröffnen "Lager" logisch "la"
  sichten "liefer" logisch "lf"
  eröffnen "bestell" logisch "be"
  aktiv "lf"
```

```
    ordnen code$; a
    leeren
    freibild
  EndeProz
```

Und schließlich noch die bereits erwähnte **freibild**-Prozedur, welche den Bildschirm leert und eine Liste der neuen Befehle anzeigt.

```
  Proz freibild
    anm ***** Maske leeren und Befehlsmenü anzeigen *****
    lokal x$
    zeigen
    zeigen "Beliebige Taste zum Weitermachen drücken"
    setzen x$=taste()
    leeren
    zeigen
    zeigen "anfrage lager eingang bestellen verkauf adieu":
    zeigen
    zeigen "Bitte gewünschte Option eingeben"
  EndeProz
```

KAPITEL 12
QL ARCHIVE ÜBERSICHT

VARIABLEN

Variablennamen können bis zu 13 Zeichen lang sein. Sie dürfen an erster Stelle keine Zahl (0-9), ansonsten jedoch beliebige Klein– und Großbuchstaben und Zahlen enthalten. Sonderzeichen sind nicht erlaubt, mit Ausnahme von "$" und ".", welchen eine spezielle Bedeutung zukommt.

Variablen, deren letztes Zeichen ein Dollarzeichen (**$**) ist, sind sog. Text– oder Zeichenketten-Variablen. Alle anderen sind numerische Variablen. Variablen, die sich auf den Inhalt eines Datensatzes beziehen, nennt man Feldvariablen. Normalerweise geht man davon aus, daß sie auf die aktuelle Datei verweisen. Dies muß jedoch nicht notwendigerweise so sein, da es auch möglich ist, den logischen Namen einer anderen eröffneten Datei anzugeben. Dieser wird mit einem Punkt dem Variablennamen vorangestellt, nach dem Schema:

Log.__Datei__Name.Feld__Name

wie etwa in **Haupt.Kontinent$**. Ein Variablenname, der einen Punkt enthält, muß sich in jedem Fall auf ein Feld in einer eröffneten Datei beziehen. Bei einem Variablennamen ohne Punkt wird ein Versuch unternommen, den Namen einer bestehenden Variablen zuzuordnen, und zwar in dieser Reihenfolge:

1. einem Feld in der aktuellen Datei
2. einer lokalen Variablen (einem Parameter in der aktuellen Prozedur, wenn vorhanden)
3. einer globalen Variablen.

Wenn keine Entsprechung gefunden wird, erfolgt eine Fehlermeldung.

SYNTAX

Unter der *Syntax* versteht man die exakte formale Strukturbeschreibung eines Befehls oder einer Funktion. Die Syntax eines Befehls spezifiziert die Parameter, die dem Befehl beigegeben werden müssen, die Reihenfolge ihrer Eingabe und gegebenenfalls die Trennzeichen zwischen den einzelnen Parametern.

Im folgenden erklären wir die Schreibweise zur Darstellung der Syntax der Archive-Programmiersprache.

AUSDRÜCKE

Ein *Ausdruck* ist eine Kombination von Konstanten, Variablen, Funktionen und Operatoren, deren Ergebnis ein einzelner Wert ist. Ein *numerischer Ausdruck* ergibt einen numerischen Wert, und ein *Zeichenkettenausdruck* ergibt einen Textwert. Beispiele:

```
3 * y * sin(x) + länge(a$)         {NUMERISCH}
"abc" + a$ + wiederh(" - ",5)      {ZEICHENKETTE}
```

Ein Ausdruck kann, wie die Beispiele zeigen, verschiedene Teilausdrücke enthalten, wobei alle Teilausdrücke vom gleichen Typ sein müssen, d.h. entweder alle vom numerischen oder alle vom Texttyp.

Syntaktische Konventionen

Die syntaktischen Konventionen für die Archive-Programmiersprache sind ganz ähnlich wie die von SuperBASIC, nämlich:

Schreibweise	Bedeutung
Kursiv	bezeichnet eine syntaktische Einheit
[ ]	Einklammerung fakultativer (wahlweiser) Elemente
* *	Einrahmung wiederholbarer Elemente
		oder
{ }	Kommentar

Syntaktische Einheiten

lit.k	Zeichenketten-Konstante
t.ausdr	Zeichenketten–(Text-) Ausdruck
n.ausdr	numerischer Ausdruck
ausdr	Ausdruck, alphanumerisch oder numerisch
ael	Anzeige-Element
var	Variablenname (alphanumerisch oder numerisch)
ldn	logischer Dateiname
dn	physischer Dateiname (max. 8 Zeichen)
pn	Prozedurname

Eine Zeichenketten-Konstante ist ein Text in einfachen oder doppelten Anführungszeichen, also 'Text' oder "Text".

Ein Zeichenketten-Ausdruck ist eine Zeichenketten-Konstante bzw. eine Kombination aus Zeichenketten-Konstanten, Textvariablen und Textfunktionen, welche als Resultat einen Zeichenketten-Wert ergeben, z.B.

```
"fred"+a$+chr(72)
```

Ein numerischer Ausdruck ist eine Zahl bzw. eine Kombination aus Zahlen, numerischen Variablen und Operatoren (+, –, *, /, usw.), welche ein numerisches Resultat ergeben, z.B.

```
(3 + x) / sin(y)
```

Es gibt vier Anzeige-Elemente: **in, tab, schrift, papier**. Die ausführliche syntaktische Beschreibung lautet:

Anzeige-Element **in** *n.ausdr, n.ausdr*
tab *n.ausdr*
schrift *n.ausdr*
papier *n.ausdr*

Die logischen Dateinamen und die Prozedurnamen unterliegen denselben Einschränkungen wie die Variablennamen. Physische Dateinamen dürfen maximal 8 Zeichen lang sein.

Hier als Beispiel einer *syntaktischen Definition* der **ordnen**-Befehl in der von uns gewählten Schreibweise:

Ordnen-Spez. = *var;* a | d
ordnen *Ordnen-Spez* *[, *Ordnen-Spez*] *

Anders ausgedrückt: **ordnen** muß von mindestens einer Ordnen-Spezifikation gefolgt sein, welche ihrerseits eine Variable enthält, die durch ein Semikolon von einem Buchstaben getrennt ist, der entweder "a" oder "d" sein muß. Außerdem können bis zu drei weitere Spezifikationen angegeben werden, solange jedes Paar durch ein Komma vom nächsten getrennt ist. Sicher stimmen Sie mit uns überein, daß die syntaktische Schreibweise sehr kompakt ist.

Aus der Syntax allein geht nicht hervor, was die Symbole für eine Bedeutung haben und was für eine Aufgabe sie erfüllen. Zu diesem Zweck müssen Sie die Kurzbeschreibung zu jedem Befehl lesen. Die Syntax gibt lediglich eine formale Beschreibung der Anzahl und des Typs der Elemente, die einem Befehl beigegeben werden können. Nicht erkennbar ist die Zahl der maximal zulässigen Wiederholungen bei wiederholbaren Elementen. **Ordnen** z.B. kann bis zu vier Paare (Variable und Buchstabe) aufnehmen.

ARCHIVE DATEIEN

Felder

Ein *Feld* ist ein Platz, der zur Aufnahme einer Zeichenkette oder einer Zahl reserviert ist.

In Archive wird jedes Feld durch einen Feldvariablen-Namen identifiziert. Welche Art von Variablen es aufnehmen kann, hängt vom Typ des Namens ab, den Sie ihm bei der Erstellung zugewiesen haben: alle Textfeldnamen enden mit einem Dollarzeichen. Ein Archive-Feld kann bis zu 255 Zeichen aufnehmen. Numerische Felder sind daran zu erkennen, daß ihre Namen am Schluß kein Dollarzeichen aufweisen. Zahlen (numerische Werte) beanspruchen immer dieselbe Anzahl von Speicherplätzen, unabhängig von ihrem Wert. Der zulässige Wertebereich ist identisch mit dem für die arithmetischen Operatoren.

Datensätze

Ein *Datensatz* ist eine Gruppe von Feldern, deren Inhalt irgendwie zusammengehört. So enthält ein typischer Datensatz ein Feld mit dem Namen, eines mit der Anschrift und ein drittes mit der Telefonnummer einer bestimmten Person. In Archive ist die Länge der *Datensätze variabel*, d.h. jeder Datensatz beansprucht genau den Platz, den er für seine Informationen braucht. Ein Archive-Datensatz kann bis zu 255 Felder enthalten.

Dateien

Eine *Datei* besteht aus einer Sammlung von ‚verwandten Datensätzen. Sie kann also zum Beispiel eine ganze Menge Datensätze mit Namen, Adressen und Telefonnummern beinhalten, die zusammen eine Kartei ergeben. Grundsätzlich kann eine Archive-Datei bis zu etwa 15000 Datensätze aufnehmen, doch in der Praxis wird dies durch die Kapazität der Microdrives auf etwa 1000 Datensätze mit je 100 Zeichen beschränkt. Eine Datei ist die fundamentale Einheit für das Speichern und Laden von Informationen auf Microdrive-Kassette. Jeder Datei muß zur eindeutigen Identifikation ein Name gegeben werden. Der physische Name wird ihr beim Anlegen zugeteilt; der logische Name kann jederzeit geändert werden.

Dateien eröffnen und schließen

Bevor Sie eine Datei lesen oder überarbeiten können, muß sie eröffnet werden. Allgemein kann man sagen, daß bei der Eröffnung einer Datei eine Kopie von der Microdrive-Kassette gemacht und in den Arbeitsspeicher geladen wird. Bei sehr langen Dateien kann es allerdings sein, daß nicht die vollständige Datei in den Speicher geladen wird.

Es ist möglich, Dateien mit **sichten** im *Nur-Lese-Modus* zu eröffnen, der, wie der Name sagt, ausschließlich Lese– jedoch keine Schreiboperationen zuläßt, d.h. keine Veränderungen am Inhalt. Daneben gibt es den Befehl **eröffnen**, der sowohl *Lese– als auch Schreiboperationen* an der Datei gestattet.

Bei jeder Eröffnung einer Datei reserviert Archive Platz für alle Feldvariablen in einem Datensatz. Diese Feldvariablen enthalten stets die Werte des aktuellen Datensatzes.

Beim Schließen einer Datei mit **schließen, neu** oder **verlassen** werden alle Änderungen, die Sie vorgenommen haben, auf die Datei übertragen, die auf Microdrive-Kassette abgespeichert ist. Die Kopie im Arbeitsspeicher wird beseitigt. Das ordnungsgemäße Schließen einer Datei ist die einzige Methode, die garantiert, daß die auf Microdrive gespeicherte Datei auch wirklich auf den neuesten Stand gebracht wird, d.h. die letzte überarbeitete Version enthält. Aus Gründen der Arbeitsspeicherökonomie empfiehlt es sich, eröffnete Dateien immer gleich zu schließen, wenn sie nicht mehr gebraucht werden.

Beim Verlassen von Archive mit dem **verlassen**-Befehl werden alle offenen Dateien automatisch geschlossen.

Computer nicht ausschalten und Kassette nicht herausnehmen, wenn sie offene Dateien enthält.

Logische Dateinamen

Beim Eröffnen einer Datei wird ihr ein *logischer Dateiname* zugewiesen, entweder ein von Ihnen freigewählter, oder der Standardname "Haupt".

Der Zweck des logischen Dateinamens ist die eindeutige Identifikation der Datei beim gleichzeitigen Bearbeiten von mehreren Dateien.

PROZEDUREN

Eine Prozedur ist ein Unterprogramm, ein Teil eines Programms, mit eigenem Namen, der mit einer Deklaration der Form

Proz *pn* [; *var* * [, *var*] *]

beginnt und von

```
EndeProz
```

abgeschlossen wird.

Prozeduren können namentlich aus anderen Programmen oder Prozeduren aufgerufen werden und können auch sich selbst aufrufen. Sie verhalten sich dann so, als ob die Prozeduranweisungen an der Stelle des Aufrufs eingesetzt worden wären.

In Archive werden **Proz** und **EndeProz** nicht direkt über die Tastatur eingegeben, sondern automatisch bei Verwendung des Programmeditors eingefügt.

DER PROGRAMM-EDITOR

Der Eintritt in den Programmeditor erfolgt mit dem **editieren**-Befehl.

Wenn sich noch keine anderen Prozeduren im Arbeitsspeicher befinden, wird sofort die Maske zum Erstellen einer neuen Prozedur präsentiert; andernfalls erscheint in der linken Spalte des Anzeigebereichs eine Liste der bereits definierten Prozeduren. Die erste davon ist hervorgehoben und ihr Code wird rechts des Balkens aufgelistet. Zur Kennzeichnung der aktuellen Zeile in der aktuellen Prozedur ist sie in einer anderen Farbe markiert.

Innerhalb des **editieren**-Befehls stehen vier Möglichkeiten zur Auswahl:

Eine Prozedur wählen
Mit **TAB** die Liste nach unten durchgehen; umgekehrte Richtung mit **SHIFT** und **TAB**. In dem Anzeigebereich rechts des Balkens erscheint stets eine Auflistung der aktuellen Prozedur.

Mit F3 das Menü der Editierbefehle anfordern
Hier kann unter vier Befehlen durch Eingabe des jeweiligen Anfangsbuchstabens gewählt werden.

L(öschen) — **ENTER** löscht die aktuelle Prozedur; das Drücken jeder anderen Taste verläßt den Befehl, ohne eine Löschoperation durchzuführen.

N(eue Proz) — Namen der neuen Prozedur eingeben und **ENTER** drücken. Falls bereits eine Prozedur gleichen Namens existiert, kann diese editiert werden.

W(egnehmen) — Entfernt Text aus der aktuellen Prozedur und speichert ihn vorübergehend in einem Puffer. Vor Aufrufen von **W** ist mit Hilfe der Pfeiltasten der Anfang (bzw. das Ende) des Zeilenblocks zur aktuellen Zeile zu machen und dann der ganze Block (mit den Pfeiltasten) für den Transfer zu markieren. Mit **ENTER** wird der Text in dem Puffer plaziert.

E(infügen) — Kopiert den Inhalt des Puffers in die aktuelle Prozedur, direkt unter die aktuelle Zeile. Der Puffer ist danach leer.

Eine Zeile wählen
Mit den aufwärts/abwärts-Pfeiltasten die gewünschte Zeile zur aktuellen Zeile machen. Die aktuelle Zeile wird hervorgehoben.

Zeileneinschübe
F4 gestattet das Einschieben von einer oder mehreren Textzeilen in eine Prozedur, direkt unter die aktuelle Zeile. Text eingeben und **ENTER** drücken. **ENTER** ohne irgendeine Angabe verläßt diese Option.

Prozedurtext korrigieren
F5 erlaubt das Editieren der aktuellen Prozedurzeile. Die betreffende Textzeile wird in die Eingabezeile kopiert, wo sie mit dem Zeileneditor korrigiert werden kann. **ENTER** überschreibt die alte Version mit der neuen.

DER BILDSCHIRM-EDITOR

Der Eintritt in den *Bildschirmeditor* erfolgt mit dem Befehl **editor**, mit dem Sie eine eigene Bildschirmmaske erzeugen bzw. die bestehende modifizieren können. Ihre fertige Maske sichern Sie auf Microdrive-Kassette. Die beiden speziellen Befehle zum Sichern und Laden sind **bsichern** und **bladen**.

Die Bildschirmmaske setzt sich aus zwei Teilen zusammen: dem festen Hintergrundtext und den Anzeigefeldern für Variablen. Mit **schirm** wird der Hintergrund angezeigt und mit **banzeige** die aktuellen Werte des Datensatzes eingefügt.

Im **Editor** gibt es zwei Optionen:

Hintergrundtext einfügen
F3 für die Bildschirm-**Editor**-Befehle drücken

Vier **Editor**-Befehle sind über **F3** zu erreichen:

L – Leeren des Bildschirms
V – Variable: reserviert Platz für Variableneintrag
T – Textfarbe: wählt aktuelle Vordergrundfarbe
P – Papier: wählt aktuelle Hintergrundfarbe

Aktiviert wird die Bildschirmmaske mit:

bladen
schirm

Wenn ein bestimmter Bildschirm aktiv ist, enthält er die aktuellen Werte seiner Variablen nach Eingabe von **banzeige** bzw. sobald die Steuerung nach Ausführung eines Programms oder eines Befehls wieder an die Tastatur übergeben wird. Die aktive Bildschirmmaske kann mit **leeren** ausgeschaltet werden; **banzeige** bleibt wirkungslos, wenn keine aktive Maske vorhanden ist. Der Arbeitsspeicher kann immer nur eine Maske gleichzeitig enthalten.

DIE BEFEHLE

Der **anzeigen**-Befehl verfügt über seine eigene Bildschirmmaske, welche jede andere überlagert.

Es folgt eine Zusammenstellung aller ARCHIVE-Befehle in alphabetischer Reihenfolge.

ÄNDERN

Gibt auf der aktuellen Bildschirmmaske die aktuellen Variablenwerte zur Änderung frei.

Syntax: **ändern**

Sie können den Inhalt beliebiger auf dem Bildschirm angezeigter Felder ändern. Es werden nicht unbedingt alle Felder eines Datensatzes in der Maske angezeigt, Höchstens an den angezeizten Feldern können Änderungen vorgenommen werden. Wenn gar keine Feldvariablen sichtbar sind, erzwingt Archive ein **anzeigen** der Datei.

Setzen Sie den Cursor mit **TAB** oder **ENTER** auf das gewünschte Feld. Variablen, die keine Datensatzfelder sind, werden übersprungen. Führen Sie dann die notwendigen Änderungen durch Überschreiben oder durch Modifizieren mit dem Zeileneditor durch. **TAB** oder **ENTER** bringt Sie zum nächsten Feld. (**SHIFT** und **TAB** für Rückwärtsbewegungen.)

Wenn Sie mit den Änderungen fertig sind, ersetzen Sie den alten Datensatz durch den neuen, indem Sie **F5** drücken. Die gleiche Wirkung hat das Drücken von **ENTER** nach Eingabe des letzten Feldes. Beim Arbeiten mit einer geordneten Datei wird der neue Datensatz automatisch richtig plaziert.

AKTIV

Macht die genannte Datei zur aktuellen Datei.

Syntax: **aktiv** *ldn*

ALLE

Durchsucht alle logischen Datensätze der Datenbank schnellstmöglich.

Syntax: **alle** [*ldn*] : ...: **endealle**

Die Durchsuchung erfolgt im allgemeinen nicht in einer bestimmten Reihenfolge. Der fakultativ angegebene logische Dateiname erzwingt die Anwendung des Befehls auf die spezifizierte eröffnete Datei; fehlt diese Angabe, wird die aktuelle Datei durchsucht.

Die Schleifenkonstruktion mit **alle...endealle** dient im wesentlichen zum Überprüfen von Dateien, weniger zum Ändern. Von einer Verwendung von **ersetzen** in einer solchen Schleife wird abgeraten, es sei denn, Sie sind ganz sicher, daß die Länge des Datensatzes unverändert bleibt. Beispielsweise ist es zulässig, einen numerischen Wert zu ändern oder einen kleingeschriebenen Text auf Großschreibung zu konvertieren. Im Zweifelsfall ist jedoch immer eine **solange**-Schleife mit der Funktion **dateiende** zu empfehlen. Beispiel:

```
anf
solange nicht dateiende()
  ...
  ersetzen
  ...
  folgesatz
  endesolange
```

ANF(ANG)

Findet den ersten Datensatz in einer genannten Datei, bzw. der aktuellen Datei, wenn auf Angabe eines logischen Dateinamens verzichtet wird.

Syntax: **anf** [*ldn*]

ANLEGEN

Legt eine Datei an, deren Inhalt durch die in der Liste spezifizierten Felder definiert wird. Die Angabe eines logischen Dateinamens ist fakultativ; fehlt ein solcher, wird von Archive die Standardvorgabe "Haupt" eingesetzt. Durch **anlegen** wird die Datei eröffnet und unter einem Dateinamen auf einer Microdrivekassette gesichert.

Syntax: **anlegen** *dn* [**logisch**: *ldn*] : *var* * [:*var*] * **endeanlegen**

ANM(ERKUNG)

Dient zum Einfügen von benutzereigenen Kommentaren innerhalb einer Prozedur. Beim Ausführen der Prozedur wird der Rest der Zeile ignoriert, d.h. nicht bearbeitet.

Syntax: **anm**

ANZEIGEN

Bewirkt Anzeigen des logischen Dateinamens der aktuellen Datei, zusammen mit einer Liste der Feldnamen mit den Feldvariablen des aktuellen Datensatzes.

Syntax: **anzeigen**

Der **anzeigen**-Befehl überlagert eine etwaige benutzerdefinierte Bildschirmmaske mit dieser Standardauflistung, welche damit auch zum aktivierten Schirmlayout gemacht wird.

AUSFÜHREN

Lädt die spezifizierte Prozedur-Datei in den Arbeitsspeicher und führt die Prozedur **start** aus.

Syntax: **ausführen** [**objekt**] *dn*

Die Bedeutung von Objekt wird beim Befehl **sichern** erläutert.

AUSZUG

Druckt die spezifizierten Felder ausgewählter Datensätze aus der aktuellen Datei in tabellarischer Form (**ser1** – Ausgabe). Bei Fehlen einer Liste mit Feldvariablen-Namen werden alle Felder gedruckt.

Syntax: **auszug** [; *var*] * [, *var*]*

Die Ausgabe kann statt an den Drucker auf eine Microdrive-Datei geschickt werden. Dazu dient der **spulen**-Befehl.

BANZEIGE

Dient im Rahmen einer Prozedur zum Erzwingen einer Anzeige der Felder im aktuellen Datensatz.

Syntax: **banzeige**

Voraussetzung ist eine aktive Bildschirmmaske (durch **schirm**, **bladen** oder **anzeigen**); ohne eine solche bleibt der Befehl wirkungslos.

BEINGABE

Wartet auf Eingabe von Werten für die Variablen der Parameterliste, entsprechend deren Reihenfolge. Alle in der Liste enthaltenen Variablen müssen in der aktiven Bildschirmmaske sichtbar sein.

Syntax: **beingabe** *var* * [, *var*]*

BLADEN

Laden und Anzeigen einer früher definierten und gespeicherten Bildschirmmaske. Aktiviert auch die Anzeige etwaiger Variablen in der Maske.

Syntax: **bladen** *dn*

Die angezeigten Werte werden jedesmal automatisch aktualisiert, wenn die Steuerung von einer Prozedur zurück an die Tastatur übergeben wird.

BSICHERN

Speichert den aktuellen Anzeigebereich als eine definierte Bildschirmmaske in Form einer benannten Datei auf einer Microdrive-Kassette.

Syntax: **bsichern** *dn*

Gespeichert wird der Hintergrundtext und eine Liste der Variablen, zusammen mit den exakten Positionen.

DRUCKEN

Bewirkt Ausgabe der Werte der vorliegenden Liste auf dem über SER1 angeschlossenen Drucker; analog zu **listen**.

Syntax: **drucken** [*ausdr* | *ael* * [; *ausdr* | *ael*] *] [;]

EDITIEREN

Ruft den Prozedur-**Editor** zum Erstellen einer neuen Prozedur bzw. zum Bearbeiten einer bereits bestehenden auf.

Syntax: **editieren**

EDITOR

Ruft den Bildschirm-**Editor** zur Definition einer neuen Bildschirmmaske auf. Ausführliche Beschreibung in Kapitel 7.

EINFÜGEN Zum Einfügen eines neuen Datensatzes in eine Datei.

Syntax: **einfügen**

In der aktuellen Bildschirmmaske werden die aktuellen Variablenwerte angezeigt, welche zur Änderung freigegeben sind. Nicht sichtbare Felder können nicht verändert werden. Falls keine Feldvariablen am Bildschirm erscheinen, erzwingt Archive ein **anzeigen** der Datei.

Als erstes setzen Sie den Cursor mit Hilfe von **TAB** oder **ENTER** auf das gewünschte Feld (Werte, die keine Felder sind, werden übersprungen). Geben Sie dann den neuen Wert ein. **TAB** oder **ENTER** bringt Sie zum nächsten Feld (**SHIFT** und **TAB** bewegen den Cursor in umgekehrter Richtung).

Nach Eingabe aller Werte bewirken Sie mit **F5** den Einschub des neuen Datensatzes in die Datei. Dieselbe Wirkung hat auch das Drücken von **ENTER**, wenn der Cursor sich im letzten Feld befindet. Felder, denen Sie keine Werte zuweisen, haben den numerischen Wert Null bzw. eine Nulltextkette. Bei einer geordneten Datei erfolgt die Einfügung an der entsprechenden Stelle; andernfalls an einer beliebigen, nichtspezifischen.

EINGABE Zur Aufforderung von Eingabe über die Tastatur an die im Befehl aufgelisteten Variablen. Jeder Variablen in einer solchen Liste kann eine Textkette vorangestellt werden, welche als Meldung zur Anforderung der Eingabe dient. Die einzelnen Elemente sind durch Semikolon voneinander zu trennen. Ein abschließender Semikolon am Ende der Zeile bewirkt, daß der Cursor nicht auf die nächste Zeile springt.

Syntax: **eingabe** [*var* | *lit.k* | *ael* *[; *var* | *lit.k* | *ael*] *][;]

Die Liste der Eingabeelemente kann auch die folgenden Anweisungen zur Cursor-Plazierung beinhalten:

in *Zeile, Spalte*
tab *Spalte*

wobei *Zeile* = *n.ausdr*
Spalte = *n.ausdr*

Die erste (in) setzt den Cursor auf die spezifizierte Zeilen/Spalten–Position, während **tab** den Cursor auf die entsprechende Spalte in der aktuellen Zeile plaziert. Falls sich der Cursor bereits rechts der **tab**-Position befindet, bleibt die Angabe wirkungslos.

Diese beiden Positionsangaben dürfen außerhalb der beiden Befehle **eingabe** und **zeigen** nicht verwendet werden. Tab kann jedoch im Befehl Drucken verwendet werden.

Nach Wunsch können auch **schrift** und **papier** als Eingabeelemente auftreten, die dann lediglich bis zum Ende der Eingabe Geltung haben. Danach werden die Farbeinstellungen wieder zurückgesetzt.

ELIMINIEREN Löscht die spezifizierte Datei auf der Microdrive-Kassette.

Syntax: **eliminieren** *dn*

Vorsicht: Diesen Befehl niemals voreilig verwenden, da einmal gelöschte Dateien unwiederbringlich verloren sind.

ENDE Findet den letzten Datensatz in der genannten Datei, bzw. in der aktuellen, wenn kein logischer Dateiname angegeben wird.

Syntax: **ende** [*ldn*]

ENDEALLE Vgl. **alle**.

ENDEANLEGEN Vgl. **anlegen**

ERSETZEN Ersetzt den aktuellen Datensatz der genannten Datei (bzw. der aktuellen, wenn kein logischer Name spezifiziert wird) durch einen Datensatz mit den aktuellen Variablenwerten.

Syntax: **ersetzen** [*ldn*]

ERÖFFNEN

Eröffnet eine Datei zum Lesen und Schreiben (Ändern, Überarbeiten). Fehlt ein logischer Name, wird "Haupt" angenommen.

Syntax: **eröffnen** *dn* [**logisch** *ldn*]

ERWEITERN

Fügt der spezifizierten (bzw. der aktuellen) Datei einen Datensatz hinzu.

Syntax: **erweitern** [*ldn*]

Der hinzugefügte Datensatz enthält die aktuellen Werte der Feldvariablen. In einer geordneten Datei wird der Datensatz an der richtigen Stelle eingefügt.

EXPORT

Speichert die benannten Felder ausgewählter Datensätze der aktuellen Archive-Datei auf Microdrivekassette, in einer Form, die den Import durch QL Abacus oder Easel gestattet.

Syntax: **export** *dn* [; *var*] * [, var] * [**quill**]

Fehlt eine Liste von Variablennamen, werden alle Felder gespeichert. Durch Hinzufügen des Parameters **quill** (durch mindestens ein Leerzeichen vom vorangehenden Variablennamen getrennt) wird die Datei auf eine Form gebracht, die zum Import durch Quill geeignet ist.

Die Exportdatei wird mit *dn* benannt; wenn Sie keinen Zusatz vorsehen, verwendet Archive den Zusatz **_exp**.

Für eine ausführlichere Beschreibung zum Import und Export verweisen wir auf den Abschnitt Information in diesem Handbuch.

FEHLER

Syntax: **fehler** *pn*[; *ausdr* *[, *ausdr*] *]

Kennzeichnet eine Prozedur im Hinblick auf Fehlerbehandlung. Jeder während der Ausführung einer solchen Prozedur (bzw. während einer durch sie aufgerufenen) auftretende Fehler bewirkt einen frühzeitigen Rücksprung in den Programmhauptteil. Die Prozedur kann die Art des Fehlers mit Hilfe von **fehlernr()** bestimmen und lesen. Diese Nummer wird bei jeder Ausführung von **fehler** auf Null gesetzt.

FINDEN

Sucht nach dem ersten Auftreten des Suchbegriffs in einem beliebigen Feld. Groß– und Kleinschreibung werden nicht berücksichtigt.

Syntax: **finden** *t.ausdr*

Zum Weiterführen der Suche **weiter** verwenden. Gegebenenfalls mit der Funktion **gefunden()** den Erfolg überprüfen.

FOLGESATZ

Springt zum nächsten Satz in der genannten Datei, bzw. in der aktuellen, wenn kein logischer Dateiname angegeben wird.

Syntax: **folgesatz** [*ldn*]

FORMATIEREN

Formatiert die Kassette im Microdrive 2 (rechts) und weist ihr den von Ihnen vorgesehenen Namen zu. Dieser Name wird bei Aufrufen von **verz** angezeigt, als Bestandteil des Dateienkatalogs.

Syntax: **formatieren** *"mdv2_Name"*

IMPORT

Liest eine aus QL Abacus oder QL Easel importierte Datei, *Name1*, und wandelt Sie in eine Archive-Datei, *Name2*, um. Wie bei **eröffnen** und **sichten** steht es Ihnen frei, der Datei einen logischen Namen zu geben.

Syntax: **import** *Name1* **auf** *Name2* [**logisch** *ldn*]

wobei *Name1* = *dn*
Name2 = *dn*

Für Einzelheiten zu den Import– und Exportfunktionen verweisen wir auf den Informationsteil.

LADEN

Lädt die spezifizierte Prozedur-Datei von Microdrive-Kassette in den Arbeitsspeicher.

Syntax: **laden** [**objekt**] *dn*

Bei Angabe von **objekt** erwartet Archive eine Datei im Binärformat statt im ASCII-Code. Siehe auch unter **sichern**.

LEEREN Leert den Anzeigebereich und schaltet die Anzeigemaske aus. Weitere Informationen unter **schirm**, **bladen** und **banzeige**.

Syntax: **leeren**

LISTEN Gibt auf dem Drucker eine Liste aller im Arbeitsspeicher befindlichen Prozeduren aus.

Syntax: **listen**

LÖSCHEN Löscht den aktuellen Datensatz aus der genannten Datei bzw. aus der aktuellen, wenn ein logischer Dateiname nicht angegeben wird.

Syntax: **löschen** [*ldn*]

Vorsicht: Diesen Befehl nicht voreilig verwenden, da einmal gelöschte Sätze unwiederbringlich verloren sind.

LOKAL Deklariert innerhalb einer Prozedur die nachfolgenden Variablen zu lokalen Variablen. Der Geltungsbereich dieser Variablen ist auf die Prozedur beschränkt, in der sie definiert werden; außerhalb sind sie unbekannt. Ihre Werte werden bei Austritt aus der Prozedur vernichtet.

Syntax: **lokal** *var* *[, *var*] *

MISCHEN Fügt die Prozeduren einer spezifizierten Programmdatei zu den bereits im Arbeitsspeicher befindlichen hinzu. Falls eine der neu eingelesenen Prozeduren den gleichen Namen aufweist wie eine im Arbeitsspeicher, wird die urspüngliche durch die neu hereingeholte Prozedur ersetzt.

Syntax: **mischen** [**objekt**] *dn*

Bei Angabe von **objekt** erfolgt das Einlesen der Datei im Binär– statt im ASCII-Format. Vgl. **sichern**.

MODUS Zum Ändern der Bildschirmbetriebsart.

Syntax: **modus** *var,var*

Die erste Variable kann 0 oder 1 sein: 0 erzeugt ein Vollbild, d.h. verbindet Steuer-, Anzeige– und Arbeitsbereich zu einer einzigen Fläche. Der Wert 1 nimmt wieder eine Dreiteilung vor.

Die zweite Variable kann einen Wert von 4, 6 oder 8 haben, entsprechend dem Bildschirmformat mit 40, 64 oder 80 Spalten pro Zeile.

Die anfängliche Einstellung beim Laden von Archive zum Arbeiten mit einem Monitor entspricht

modus 1,8

NEU Löscht alle Daten aus dem Arbeitsspeicher und bewirkt einen Neustart. Etwaige eröffnete Dateien werden geschlossen. (Der Befehl hat nicht etwa das Löschen von auf Microdrive gespeicherten Dateien zur Folge.)

Syntax: **neu**

NORMAL Leitet alle folgenden Druckausgaben mit **listen** und **drucken** an den angeschlossenen Drucker und annulliert die Wirkung von **spulen**.

Syntax: **normal**

ORDNEN Sortiert die Datensätze der Datei nach dem Inhalt der spezifizierten Felder.

Syntax: **ordnen** *Ordnen-Spez.* *[, *Ordnen-Spez.*] *

wobei *Ordnen-Spez.* = *var*; **a** | **d**

Das erste Feld in der Liste ist das primäre Sortierfeld. Datensätze, deren Inhalt bezüglich dieses primären Sortierfelds identisch ist, werden zusätzlich nach dem zweiten Feld in der Liste sortiert (wenn eine solches vorhanden ist) usw. Für jedes Sortierfeld ist eine Richtung anzugeben, d.h. a für eine aufsteigende, d für eine absteigende Folge.

Der Sortiervorgang berücksichtigt nur die ersten 8 Zeichen eines Textfeldes, und es dürfen maximal vier Sortierfelder vorgesehen werden.

ORTEN

Ermittelt in einer sortierten Datei den ersten Datensatz, dessen Feldinhalt mit dem Suchbegriff übereinstimmt.

Syntax: **orten** *ausdr* * [*,ausdr*] *

Das Auffinden des Datensatzes geht erheblich schneller als bei **finden**, doch wird eine **geordnete Datei vorausgesetzt**. Jeder Ausdruck muß sich explizit auf den Inhalt eines bestimmten Sortierfelds beziehen. Bei Textfeldern wird Groß– und Kleinschreibung mitberücksichtigt.

Falls die Datei nach mehr als einem Sortierfeld geordnet worden ist, sind mehrere Ausdrücke zulässig (je einer pro Sortierfeld). Mehrere Ausdrücke sind durch Kommas voneinander zu trennen und müssen sich auf die Felder beziehen, nach denen der Sortiervorgang erfolgte, d.h. ihre Reihenfolge muß identisch sein mit der beim Ordnen verwendeten, z.B.

```
ordnen Tier$ ; a , Gewicht ; a
orten "Elefant" , 2000
```

Damit wird der erste Datensatz gefunden, dessen Textfeld "Tier$" mit der Zeichenkette "Elefant" beginnt und ein Gewicht von mindestens 2000 aufweist.

Wenn keine exakte Entsprechung gefunden wird, greift **orten** den ersten Datensatz heraus, dessen Inhalt die spezifizierten Werte im Sinne der Sortierreihenfolge "übersteigt" (d.h. "danach kommt"). Bei absteigender Folge kommt z.B. "d" nach "e".

PAPIER

Setzt die Hintergrundfarbe für nachfolgenden Text auf die durch den Wert spezifizierte Farbe.

Syntax: **papier** *n.ausdr*

Die möglichen Farben sind: 0 und 1 schwarz
2 und 3 rot
4 und 5 grün
6 und 7 weiß

Falls der Ausdruck einen Wert über 7 ergibt, gilt der Divisionsrest nach Teilung durch 8, d.h. **papier** 11 ist gleich **papier** 3 – beide setzen die Farbe auf rot.

POSITION

Macht den Satz mit der im Ausdruck spezifizierten Nummer zum aktuellen Datensatz.

Syntax: **position** *n.ausdr*

PROTOKOLL

Schaltet den Protokollmodus ein und aus.

Syntax: **protokoll**

Eingabe von **protokoll** schaltet das Protokoll ein, d.h. im Arbeitsbereich des Bildschirms wird der Programmablauf zeilenweise angezeigt. Zum Anhalten die Leertaste gedrückt halten; zum Fortsetzen der Ablaufverfolgung die Leertaste wieder freigeben. Zum Ausschalten des Protokolls ein zweites Mal **protokoll** eingeben. Ist **protokoll** eingeschaltet, so kann es während des Programmlaufs mit der Taste **F3** vorübergehend ein- und ausgeschaltet werden.

RESERVE

Erstellt eine Kopie der spezifizierten Datei. Es empfiehlt sich, von allen Dateien Reservekopien anzulegen, um sich gegen Datenverlust durch versehentliches Überschreiben oder Beschädigung zu schützen.

Syntax: **reserve** *alt.dn* **auf** *neu.dn*

RÜCKKEHR

Wird innerhalb einer Prozedur zum sofortigen Abbruch der gerade in der Durchführung befindlichen Prozedur und zum Rücksprung in die aufrufende Prozedur verwendet.

Syntax: **rückkehr**

RÜCKSETZEN

Stellt alle Datensätze der aktuellen Datei wieder her, welche infolge eines **wählen**-Vorgangs extrahiert wurden. Eine etwaige Ordnung der Datenbank geht damit verloren.

Syntax: **rücksetzen**

SCHIRM Zeigt die vorher mit **bladen** eingelesene Bildschirmmaske an. Fehlt eine solche, bleibt der Befehl wirkungslos. Es werden keine Variablenwerte angezeigt.

Syntax: **schirm**

SCHLIEßEN Schließt die spezifizierte Datei bzw. die aktuelle, wenn kein logischer Dateiname angegeben wird.

Syntax: **schließen** [*ldn*]

SCHRIFT Setzt die Vordergrund– (Zeichen-) Farbe des nachfolgenden Texts auf die durch den Wert des Ausdrucks spezifizierte Farbe.

Syntax: **schrift** *n.ausdr*

Die verfügbaren Farben sind: 0 und 1 schwarz
2 und 3 rot
4 und 5 grün
6 und 7 weiß

Wenn der Wert des Ausdrucks höher als 7 ist, gilt der Divisionsrest nach Teilung durch 8. So ist z.B. **schrift** 9 identisch mit **schrift** 1 – beide setzen die Farbe auf schwarz. Bei Verwendung von **schrift** in einem **zeigen**-Befehl gilt die Einstellung nur für die Dauer des betreffenden Befehls.

SETZEN Weist einer Variablen einen Wert zu (wie LET in SuperBASIC).

Syntax: **setzen** *var = ausdr*

SICHERN Speichert alle gegenwärtig im Arbeitsspeicher befindlichen Prozeduren in Form einer einzigen benannten Datei auf eine Microdrive-Kassette.

Syntax: **sichern** [**objekt**] *dn*

Bei Angabe von **objekt** speichert Archive die Datei in Binär– statt in ASCII-Format. Damit erübrigt sich die Umwandlung des Programms in ASCII-Zeichen, so daß der Vorgang beschleunigt wird. Ein solches Programm kann geladen, ausgeführt und gemischt werden, indem das fakultative Objekt im Befehl mit angegeben wird. Auch hier wird durch das Einsparen der Umwandlung eine Beschleunigung erwirkt. Die Dateien werden mit dem speziellen Nachnamen _pro (statt _prg) versehen.

Derartige Objektprogramme können auch in einer Form gesichert werden, die gegen Überprüfung und Veränderung geschützt ist. Zu diesem Zweck wird statt **objekt** das fakultative **schützen** verwendet. Ein auf diese Art gespeichertes Programm kann nur unter Angabe von **objekt** geladen, ausgeführt oder gemischt werden.

Geschützte Programme können nicht aufgelistet, editiert oder gesichert werden. Beim Mischen mit einem beliebigen anderen Programm wird die Kombination ebenfalls in den geschützten Zustand versetzt, welcher nur mit dem Befehl **neu** aufgehoben werden kann.

Das Sichern einer geschützten Version betrifft nicht die Programmkopie im Arbeitsspeicher, welche nach wie vor gelistet, editiert oder gesichert werden kann.

SICHTEN Eröffnet die genannte Datei nur für Leseoperationen. Ohne ausdrückliche Angabe eines logischen Namens wird "Haupt" angenommen.

Syntax: **sichten** *dn* [**logisch** *ldn*]

SOLANGE Führt die Programmanweisungen in der Schleife, die durch **solange** und **endesolange** gebildet wird, solange aus, wie der Wert des Ausdrucks wahr ist, d.h. ungleich Null.

Syntax: **solange** *n.ausdr: ....:* **endesolange**

SPULEN Leitet alle folgenden Ausgaben (mit **drucken**, **listen** und **auszug**) an die genannte Datei bzw. an den Bildschirm statt an den Drucker.

Syntax: **spulen** *dn* [**export** | *auszug*]
oder:
spulen schirm

Die Umleitung der Ausgabe an eine Datei erfolgt über den installierten Druckertreiber, so daß alle für Ihren Drucker erforderlichen Steuercodes mit enthalten sind.

Bei Angabe des fakultativen Zusatzes **export** stellt Archive sicher, daß die Datei nur druckbare ASCII-Codes und Codes für Wagenrücklauf und Zeilenvorschub enthält. Eine solche Datei kann durch Quill importiert werden.

Die fakultative Angabe **auszug** gestattet eine Textübertragung direkt an die Datei, ohne Verarbeitung durch den Druckertreiber. Auf diese Weise werden alle ASCII-Codes (einschließlich Steuerzeichen) an die Datei gesandt.

Bei Fehlen eines ausdrücklichen Datei-Zusatzes nimmt Archive _lis (bzw. _exp oder _dmp) an.

STOPP

Bricht die Ausführung aller Prozeduren ab und übergibt die Steuerung wieder an die Tastatur.

Syntax: **stopp**

SUCHEN

Durchsucht die aktuelle Datei ab dem Anfang nach dem ersten Datensatz, für den der spezifizierte Ausdruck wahr ist. Dieser Datensatz wird dann zum aktuellen.

Syntax: **suchen** *n.ausdr*

VERLASSEN

Schließt alle Dateien und kehrt zu SuperBASIC zurück.

Syntax: **verlassen**

VERZ

Gibt einen Katalog aller Dateien auf einer Microdrivekassette aus.

Syntax: **verz** [*laufwerk*]

Das gewünschte Laufwerk kann mit **mdv1_** oder **mdv2_** spezifiziert werden. Fehlt eine solche Angabe, nimmt Archive Microdrive 2 an.

Vor Anzeigen der Dateienliste erscheint der Datenträgername, den Sie der Kassette beim Formatieren gegeben haben, am Bildschirm.

WÄHLEN

Durchsucht die gesamte Datenbank und greift die Datensätze heraus, für die der spezifizierte Ausdruck wahr ist. Die Datei verhält sich dann so, als ob nur noch die ausgewählten Datensätze vorhanden wären.

Syntax: **wählen** *n.ausdr*

Die bei dem Prozeß eliminierten Datensätze lassen sich mit **rücksetzen** wiederherstellen.

WEITER

Zum Fortführen der vorangehenden **suchen** – oder **finden**-Operation, ausgehend vom nächsten Datensatz in der aktuellen Datei.

Syntax: **weiter**

WENN

Überläßt der spezifizierten Bedingung die Entscheidung über den weiteren Programmverlauf.

Syntax: **wenn** *n.ausdr* : ... [: **sonst** : ...] : **endewenn**

Ohne **sonst**:
Wenn der Ausdruck ungleich Null (also wahr) ist, werden die nachfolgenden Anweisungen ausgeführt; andernfalls erfolgt ein Sprung zu der Programmanweisung nach **endewenn**.

Mit **sonst**:
Wenn der numerische Ausdruck ungleich Null ist, gelangen die Anweisungen zwischen **wenn** und **sonst** zur Ausführung; wenn er Null ist, jene zwischen **sonst** und **endewenn**. In beiden Fällen geht es danach mit den Programmanweisungen nach **endewenn** weiter.

ZEIGEN

Zeigt die Werte der nachfolgenden Listenelemente an, welche durch Semikolon voneinander getrennt sein müssen. Ein Semikolon am Ende der Liste bewirkt, daß der Cursor nicht auf die nächste Zeile springt. Vergleichen Sie auch unter **drucken**.

Syntax: **zeigen** [*ausdr* | *ael*] * [; *ausdr* | *ael*] *] [;]

ZURÜCK Springt auf den vorangehenden Datensatz in der spezifizierten Datei bzw. in der aktuellen Datei (wenn kein logischer Dateiname angegeben wird) zurück und macht diesen zum aktuellen Satz.

Syntax: **zurück** [*ldn*]

FUNKTIONEN

Stellen Sie sich eine Funktion am besten als eine Art Rezept vor, das mit einem oder mehreren Werten, den sog. *Argumenten*, eine bestimmte Operation ausführt und ein Ergebnis liefert, den *Funktionswert*.

In Archive gibt es *Funktionen*, die ein, zwei oder drei Argumente übernehmen, und auch solche, die ohne ein Argument auskommen. Die Argumente werden direkt nach der *Funktion* in Klammer angegeben. Zwischen dem *Funktionsnamen* und der Eröffnungsklammer darf kein Leerzeichen gelassen werden; hingegen sind Leerzeichen zwischen den Werten in der Klammer gestattet. Mehrere Argumente sind durch Kommas voneinander zu trennen. Alle *Funktionen* müssen von einem Klammerpaar gefolgt werden, auch wenn sie keine Argumente enthalten. Dies hilft, eine *Funktion* als solche zu identifizieren und Verwechslungen zwischen Variablen und *Funktionen* zu vermeiden, falls sie den gleichen Namen haben.

Es folgt eine alphabetische Zusammenstellung aller in Archive verfügbaren Funktionen.

ABS(*n.ausdr*) liefert den absoluten Wert ihres Arguments (d.h. den Wert ohne Berücksichtigung eines etwaigen Minuszeichens)

ALLG(*wert,breite*)

wert:= n.ausdr
breite:= n.ausdr

Wandelt den vorliegenden numerischen Wert in die entsprechende Textkette um. Der Text erhält allgemeines Format und wird rechtsbündig in ein Feld von *breite* Zeichen gesetzt. Beispiel:

allg(1.23e1,10)

ergibt den Text "   12.3" (mit 6 führenden Leerzeichen).

ANZAHL(*ldn*) Liefert die Anzahl der Datensätze in der genannten bzw. der aktuellen Datei.

BOG(*n*) Wandelt einen in Grad ausgedrückten Winkel in Bogenmaß um.

CODE(*t.ausdr*) Liefert den ASCII-Code des ersten Zeichens im angegebenen Textargument.

DATEIENDE(*ldn*)

Liefert als Ergebnis einen Wert, der anzeigt, ob ein Versuch unternommen wurde, über das Dateiende der aktuellen Datei (bzw. der spezifizierten Datei) hinauszulesen. Wurde das Dateiende entdeckt, ist das Ergebnis 1; andernfalls 0.

DATUM(*n.ausdr*)

Gibt das aktuelle Datum als Zeichenkette aus, wobei drei Formen möglich sind:

n.ausdr	Datumszeichenkette
0	"JJJJ/MM/TT"
1	"TT/MM/JJJJ"
2	"MM/TT/JJJJ"

Voraussetzung ist das Setzen der internen System-Uhr (vgl. SDATE in Abschnitt Befehle in SuperBASIC).

DEZ(*wert,DSt,breite*)

wert:= (n.ausdr)
DSt:= (n.ausdr)
breite:= (n.ausdr)

Wandelt den gegebenen numerischen Wert in die entsprechende Textkette um. Der Text wird im Dezimalformat mit *DSt* Dezimalstellen rechtsbündig in einem Feld von *breite* Zeichen ausgegeben, z.B.:

dez(1.23e1,3,10)

liefert als Resultat den Text "   12.300" (mit 4 führenden Leerzeichen).

EXP(*n.ausdr*) Liefert den Wert von e (ca. 2,718) hoch n, wobei das Resultat verfälscht ist, wenn n außerhalb des Bereichs −87 bis +88 liegt, weil dies den numerischen Bereich von Archive übersteigt.

FEHLERNR() Liefert die Nummer des zuletzt aufgetretenen Fehlers. Fehlernummer 0 bedeutet, daß keine Fehler entdeckt wurden. Die Nummer ist identisch mit der durch Archive anläßlich einer Fehlermeldung ausgegebenen.

FELDNAME(*n.ausdr[,ldn]*)
Gibt den Namen des spezifizierten Feldes im aktuellen Datensatz (der angegebenen bzw. der aktuellen Datei). Hinweis: **feldname(0)** liefert den Namen des ersten Feldes.

FELDTYP(*n.ausdr[, ldn]*)
Gibt den Typ des spezifizierten Felds im aktuellen Datensatz (der angegebenen bzw. der aktuellen Datei). Hinweis: **feldtyp(0)** liefert den Typ des ersten Felds.

Der Wert von Feldtyp ist 0 bei einem numerischen Feld, sonst ist er 1.

FELDWERT(*n.ausdr[, ldn]*)
Liefert den Wert des spezifizierten Felds im aktuellen Datensatz (der angegebenen bzw. der aktuellen Datei). Hinweis: **feldwert(0)** liefert den Wert des ersten Felds.

FELDZAHL(*[ldn]*)
Liefert die Anzahl der Felder eines Datensatzes in der benannten oder der aktuellen Datei.

GANZZAHL(*n.ausdr*)
Liefert den ganzzahligen Teil des Arguments und schneidet etwaige Dezimalstellen ab. Die Rundung erfolgt immer in Richtung 0, z.B.:

```
ganzzahl(3.7)      {liefert 3}
ganzzahl(-4.8)     {liefert -4}
```

GEFUNDEN() Gibt das Resultat 1 aus, wenn die letzte Suchoperation mit **suchen** oder **finden** erfolgreich war; andernfalls 0.

GRAD(*n.ausdr*) Zur Umwandlung von Bogenmaß in Grad.

GROß(*t.ausdr*) Wandelt die angegebene Zeichenkette in Großbuchstaben um.

KETTE(*n,Typ,Dst*)
n = n.ausdr
Typ = n.ausdr
Dst = n.ausdr

Wandelt eine Zahl n in die entsprechende Zeichenkette um. Der zweite Parameter, Typ, kennzeichnet das Format der umgewandelten Zeichenkette:

0 = Dezimal (Gleitkomma)
1 = Exponential
2 = Ganze Zahl
3 = Allgemein

Der dritte Parameter, Dst, spezifiziert die Anzahl der Nachkommastellen in der umgewandelten Kette und muß immer angegeben werden, auch wenn die Formate Ganze Zahl und Allgemein davon keinen Gebrauch machen.

Beispiele:

```
setzen a$=kette(12.3456,0,2)
     {gibt a$ den Wert "12.35"}
setzen a$=kette(12.3456,1,4)
     {gibt a$ den Wert "1.2346e1"}
```

KLEIN(*t.ausdr*) Wandelt die angegebene Zeichenkette in Kleinschreibung um.

KOS(*n.ausdr*) Liefert den Kosinus des (in Bogenmaß) gegebenen Winkels.

LÄNGE(*t.ausdr*) Gibt die Anzahl der Zeichen im spezifizierten Text an.

LN(*n.ausdr*) Gibt den natürlichen Logarithmus (zur Basis e) des Arguments n. Ein Null– oder Negativwert von n provoziert eine Fehlermeldung, da keine Logarithmen für diesen Wertebereich definiert sind.

MONAT(*n.ausdr*)

Gibt den dem numerischen Argument entsprechenden Monatsnamen als Textwert an.

Beispielsweise liefert **monat(3)** die Zeichenkette "März".

Bei Eingabe eines Arguments, das größer als 12 ist, wird n durch 12 geteilt, und der Rest der Division ist ausschlaggebend: **monat(13)** und **monat(1)** ergeben beide "Januar".

NUM(*wert,breite*)

wert:= n.ausdr
breite:= n.ausdr

Wandelt das numerische Argument in die entsprechende Textkette um. Der Text wird rechtsbündig in einem Feld von *breite* Zeichen angeordnet, z.B.

num(1.23e1,10) liefert den Textwert "   12" (8 führende Leerzeichen).

NUMWERT(*t.ausdr*)

Wandelt das Textargument in einen numerischen Wert um. Das erste nichtnumerische Zeichen des Arguments beendet die Umwandlung. Ist das erste Zeichen des Textes keine Ziffer oder kein Dezimalpunkt, so liefert die Funktion den Wert Null.

PI() Liefert den Wert der mathematischen Konstanten π

QWURZEL(*n.ausdr*)

Gibt die Quadratwurzel ihres Arguments aus, welches keine negative Zahl sein darf.

SATZNR(*ldn*) Liefert die laufende Nummer des aktuellen Datensatzes in der angegebenen bzw. der aktuellen Datei. Der erste Satz einer Datei hat die laufende Nummer Null.

SIN(*n.ausdr*) Liefert den Sinus des Winkels n (in Bogenmaß).

SPEICHER() Gibt den Wert des noch verfügbaren Speicherplatz aus.

TAGE(*t.ausdr*) Liefert die seit dem 1. Januar 1583 verflossenen Tage bis zu einem als Textausdruck in der Form "JJJJ/MM/TT" gegebenen Datum. Die Umwandlung geht vom Gregorianischen Kalender aus und gilt folglich nur für Daten nach 1582.

TAN(*n.ausdr*) Liefert den Tangens des angegebenen Winkels (in Bogenmaß).

TASTE() Wartet einen Tastenanschlag ab und liefert als Wert das Einzelzeichen, welches der gedrückten Taste entspricht.

TEILKETTE(*Haupt,Teil*)

Haupt = t.ausdr
Teil = t.ausdr

Findet das erste Auftreten des Teilbegriffs in der Hauptzeichenkette und liefert die Position des ersten Zeichens. Wird keine Entsprechung gefunden, ist das Ergebnis Null. Die Schreibung (groß oder klein) wird berücksichtigt.

```
teilkette ("Januar","Jan")      {Ergebnis: 1}
teilkette ("Januar","an")       {Ergebnis: 2}
teilkette ("Januar","AN")       {Ergebnis: 0}
```

TIPPEN() Liefert das Einzelzeichen, welches der Taste entspricht, die beim Aufruf der Funktion angeschlagen wurde. Wenn kein Tastendruck erfolgt, wartet die Funktion nicht, sondern gibt eine Nullkette (" ") aus.

VORZ(*n.ausdr*) Liefert +1 bei einem positiven, –1 bei einem negativen und 0 bei einem Nullargument.

WERT(*t.ausdr*) Liefert den Wert der Variablen, deren Name das Argument *t.ausdr* bildet. Beispiel:

```
setzen a$="länge"
setzen länge=15
zeigen wert(a$)
```

gibt den Wert 15 aus.

WIEDERH(*t.ausdr,n.ausdr*)
Diese Funktion liefert eine Kette, die eine Anzahl Kopien des ersten Zeichens von *t.ausdr* enthält. Die maximale Länge der Kette beträgt 255 Zeichen. Beispiele:

```
zeigen wiederh("*",5)
    {gibt 5 Sternchen am Bildschirm aus}
zeigen wiederh("abc",3)
    {gibt die Kette "aaa" aus}
```

WTN(*n.ausdr*) Gibt den Winkel (Bogenmaß) aus, dessen Tangens *n.ausdr* ist (Arcustangens).

ZCHN(*n.ausdr*) Liefert das ASCII-Zeichen, dessen Code *n.ausdr* ist. Zeichen mit ASCII-Codes unter 32 werden nur an den Drucker geleitet, wenn ihnen eine ASCII-Null vorangeht, z.B. bewirkt

```
drucken zchn(0)+zchn(13)
```

die Übermittlung des Steuerzeichens für einen Wagenrücklauf an den Drucker.

Dies erweist sich als praktisch, wenn Ihr Drucker spezielle Steuersequenzen zur Erzeugung von drucktechnischen Sonderfunktionen fordert. Diese entnehmen Sie dem Handbuch zu Ihrem Drucker.

ZEIT() Liefert als Textwert die Uhrzeit im Format "hh:mm:ss". Vorausssetzung ist das Setzen der internen System-Uhr, beschrieben unter dem SuperBASIC Befehl sdate.

FEHLER

Bei Auffinden eines Fehlers in einem über die Tastatur eingegebenen Befehl oder innerhalb einer Prozedur gibt Archive eine Fehlernummer, zusammen mit einer kurzen Nachricht, an. Typische Fehlerzustände sind beispielsweise:

Versuch einer Division durch Null
Wenn ohne entsprechendes **endewenn**
Übergabe der falschen Anzahl Parameter an eine Prozedur.

Wenn der Fehler bei der Eingabe über die Tastatur geschieht, bleibt die Anweisung im Arbeitsbereich sichtbar. Mit **F5** können Sie den Text zurückholen und ihn mit dem Zeileneditor entsprechend korrigieren. Drücken von **ENTER** führt dann die korrigierte Anweisung aus.

Tritt der Fehler im Rahmen eines Programms (einer Prozedur) auf, zeigt ARCHIVE den betreffenden Prozedurnamen, zusammen mit der fehlerhaften Anweisungszeile. Benutzen Sie den Programmeditor, um den Fehler zu beheben.

Bei Verwendung des **fehler**-Befehls meldet Archive keine fehler, die innerhalb von mit **fehler** markierten Prozeduren auftreten. Es bleibt Ihnen überlassen, diese Fehler nach Gutdünken zu behandeln bzw. sie zu ignorieren. Welcher Fehler entdeckt wurde, läßt sich anhand des durch **fehlernr()** ausgegebenen Wertes nachprüfen. Dieser ist identisch mit der Nummer, die normalerweise durch Archive angezeigt wird.

Die folgende Aufstellung zeigt die Archive-Fehlernummern mit den dazugehörigen Meldungen und einer typischen Anweisung, die den Fehler verursachen würde. Die Fehlermeldungen können nicht zur exakten Identifizierung des Fehlers dienen, geben jedoch Hinweise auf die Art des Fehlers, um den es sich handeln muß.

Fehlermeldungen, die nicht anhand eines kurzen Beispiels provoziert werden können, sind mit einem Sternchen gekennzeichnet und werden im Anschluß an die Zusammenstellung beschrieben.

Nr.	Fehlermeldung	Beispiel
0	Kein Fehler	
1	Befehl nicht erkannt	AICHTEN
2	Ende der Anweisung erwartet	`setzen x=3 setzen x=4`
3	Variablenname erwartet	`setzen 31=x`
4	Unbekannter Druckposten	`zeigen anlegen`
5	Falscher Datentyp	* (1)
6	Numerischer Ausdruck erwartet	`setzen x="Moritz"`
7	Zeichenketten-Ausdruck erwartet	`setzen x$=4`
8	Variable nicht gefunden	`setzen x=qq` (qq nicht definiert)
10	Fehlendes Trennzeichen	`zeigen in 5`
11	Name zu lang	`setzen diesenganzenlangennamen=4`
12	Duplizierter Name	`anlegen:n$:n$:endeanlegen`
13	Zeichenketten-Konstante erwartet	* (2)
14	EndeProz fehlt	* (3)
15	Falsch formulierte Proz-Anweisung	* (3)
16	Vorzeitiges Anweisungsende	`anlegen"test":endeanlegen`
17	Programmstruktur-Fehler	* (4)
18	Zu viele Zahlen	* (5)
50	Anführungszeichen fehlt	`setzen x$="Moritz`
51	Fehlender Exponent nach "E"	`setzen x=1.2E`
52	Zahlen zu groß	`setzen x=1.2E100`
53	Unbekanntes Symbol	`setzen x=%`
70	Berechnungs-Syntaxfehler	`setzen x=3+`
71	Nicht übereinstimmende Klammer	`setzen x=(3+5)/7)`
72	Keine Typenübereinstimmung	`setzen x$="Fritz"+3`
74	Falsche Zahl von Argumenten	`setzen x$=kette(1,2)`
75	Zeichenkette zu lang	`setzen x$=wiederh("*",256)`
76	Division durch Null	`setzen a=0: setzen x=5/a`
77	Falsch formulierte Funktionsargumente	`setzen x$=qwurzel(-4)`
78	Zeichenkettenindex-Fehler	`setzen x$="Fritz"(bis 97)`
80	Nicht genug Speicher	* (6)
90	Kein Platz zur Eröffnung einer Datei	* (7)
91	Unvollständige Dateiübertragung	* (8)
93	Außerhalb des Bereichs	`zeigen in 100,100;37`
94	Datei nicht eröffnet	`erweitern` (ohne vorheriges Eröffnen)
100	Kann Datei nicht eröffnen	`Sichten "xxx"` (wenn keine Datei dieses Namens existiert)
101	Schreiben auf Nur-Lese-Datei	`Sichten "Namen":Einfügen`
103	Falscher Dateityp	`Bladen"Namen"` (Datei)
104	Falsch formulierter Dateiname	`Sichern "3Test"`
105	Dateilesefehler	* (9)

Anmerkungen

1) Die naheliegendste Ursache für Fehler 5 ist die versehentliche Eingabe von Text, wenn Zahlen erwartet werden, z.B. bei einer **eingabe**-Anweisung wie

```
eingabe x
```

2) Fehler 13 tritt beispielsweise während eines Importvorgangs auf, wenn Sie eine Datei hereinholen wollen, ohne vorher die erforderlichen **export**-Befehle verwendet zu haben. Konkret heißt es, daß Archive eine Zahl, einen numerischen oder einen Textausdruck an einer Stelle gefunden hat, wo es eine Konstante erwartet. In den meisten anderen Situationen, wo numerische statt alphanumerische Daten (bzw. alphanumerische statt numerische) entdeckt werden, handelt es sich um die Fehler 7 oder 8.

3) Die Fehler 14 und 15 dürften im allgemeinen nicht vorkommen. Sie weisen darauf hin, daß Archive ein fehlendes **EndeProz** angetroffen hat oder einen Fehler in der Struktur einer **Proz**–Anweisung. Diese Art von Fehler kann eigentlich nur auftreten, wenn zum Schreiben der Prozedur nicht der Archive-eigene Editor verwendet wird.

4) Fehler 17 signalisiert in den meisten Fällen das Auffinden eines **alle**, **wenn** oder **solange** ohne entsprechenden Schleifenabschluß mit **endealle**, **endewenn** oder **endesolange** innerhalb einer Prozedur. Weitere Ursachen für den gleichen Fehler sind das Schreiben von **endeproz** im Rahmen eines anderen Programms oder die Eingabe von **rückkehr** direkt über die Tastatur.

5) Fehler 18 quittiert den Versuch, mehr Zahlen einzugeben, als in den für Eingabe reservierten Speicherbereich passen. Der Fehler kann bei einer Eingabe über die Tastatur auftreten oder beim Laden eines Programms, welches eine Prozedur einschließt, die eine Zeile mit vielen Zahlen enthält. Wieviele Zahlen zulässig sind, hängt von den Umständen ab; meist liegt die obere Grenze bei 15 bis 20 Zahlen, so daß diese Fehlersituation eher rar sein dürfte.

6) Fehler 80 sollte nur bei einem sehr umfangreichen Programm auftreten. Die Größe einer normalen Datei wird nicht durch die Speicherkapazität begrenzt, da jeweils nur ein bestimmter Teil im Arbeitsspeicher zugänglich ist. Bei Ausgabe dieser Fehlermeldung gibt es keine andere Möglichkeit, als das Programm zu verkürzen, ehe Sie weiterarbeiten. Bei Bedarf ist das Programm in mehrere Abschnitte zu unterteilen und in verschiedenen Dateien zu speichern, die dann mit **mischen** nacheinander geladen werden können. Diese Methode setzt allerdings einige Programmierkenntnisse voraus.

7) Fehler 90 tritt auf, wenn der für interne Informationen über die im Speicher befindlichen Dateien reservierte Speicherbereich überfüllt ist. Dies geht nicht notwendigerweise einher mit der verfügbaren Speicherkapazität, d.h. es kann selbst dann vorkommen, wenn der durch **speicher()** ausgegebene Wert noch weit von Null entfernt ist.

8) Fehler 91 weist auf ein Versagen beim Laden oder Sichern einer Datei hin, was unter Umständen einen Verlust der Daten oder eine Beschädigung der Microdrive-Kassette bedeutet.

9) Fehler 105 tritt auf, wenn einige Daten in einer Datei ein falsches Format, eine fehlerhafte Reihenfolge oder eine Beschädigung aufweisen. Ursachen für diese Fehlerzustände sind die Konstruktion einer eigenen Importdatei oder das Schreiben einer eigenen Programmdatei ohne Verwendung des Archive-eigenen Programmeditors (fortgeschrittene Anwendungen).

KAPITEL 1
WAS IST QL ARCHIVE . . . 1

KAPITEL 2
FANGEN SIE AN . . . 2
QL ARCHIVE LADEN . . . 2
ANZEIGEFORMAT . . . 2
Anzeige- und Arbeitsbereich . . . 2
Der Steuerbereich . . . 3
UMGANG MIT BEFEHLEN . . . 3
DER MODUS-BEFEHL . . . 4

KAPITEL 3
QL ARCHIVE DATEIEN . . . 5
DATEIEN, DATENSÄTZE UND FELDER . . . 5

KAPITEL 4
EINBLICK IN DATEIEN . . . 6
DATENSATZ ANZEIGEN . . . 10
WEITERE DATENSÄTZE . . . 6
DURCHSUCHEN EINER DATEI . . . 7
Finden . . . 7
Weiter . . . 7
Suchen . . . 7
Wählen . . . 7
DATEIEN SORTIEREN . . . 7
ORTEN . . . 8
DATEI SCHLIESSEN . . . 8

KAPITEL 5
DATEIEN BEARBEITEN . . . 10
EINFÜGEN . . . 10
LÖSCHEN . . . 10
DATENSÄTZE BEARBEITEN . . . 10
Ändern . . . 11
Ersetzen . . . 11
DATEIEN SCHLIESSEN . . . 11

KAPITEL 6
DATEIEN ANLEGEN . . . 12
ANLEGEN . . . 12
DATENSÄTZE EINFÜGEN . . . 12

KAPITEL 7
BILDSCHIRM-MASKEN . . . 14
MASKEN AUFBAUEN . . . 14
BILDSCHIRM-EDITORBEFEHLE . . . 14
Variable (V) . . . 14
Schrift (S) . . . 15
Papier (P) . . . 15
MASKE AKTIVIEREN . . . 15
MASKE SICHERN UND LADEN . . . 15
DER BEFEHL BANZEIGE . . . 15
DER BEFEHL ANZEIGEN . . . 16

KAPITEL 8
PROZEDUREN . . . 17
PROZEDUREN ERSTELLEN . . . 17
PROZEDUREN AUFLISTEN UND DRUCKEN 18
PROZEDUREN SICHERN UND LADEN . . . 18
PROZEDUREN ÜBERPRÜFEN . . . 18

KAPITEL 9
DER PROGRAMM-EDITOR . . . 20
EDITIEREN . . . 20
EDITIERBEFEHLE . . . 20
Neue Prozedur (N) . . . 20
Löschen Prozedur (L) . . . 21
Wegnehmen (Zeilen) (W) . . . 21
Einsetzen (Zeilen) (E) . . . 21

KAPITEL 10
PROGRAMMIEREN MIT ARCHIVE . . . 22
VERSANDLISTE . . . 22
Ausgaben . . . 22
Einfügen . . . 23
Löschen . . . 23
Zahlungen . . . 23
Änderungen . . . 25
PARAMETER . . . 25
Adreß-Etiketten . . . 26
Programm verlassen . . . 27
FEHLER . . . 27
PROGRAMM AUSFÜHREN . . . 28
LOKALE VARIABLEN . . . 29
EINGABE ANFORDERN . . . 30
PAUSE . . . 30
DATENEINGABE . . . 31
Text . . . 31
Zahlen . . . 31

KAPITEL 11
ARBEITEN MIT MEHREREN DATEIEN . . . 32
LOGISCHE DATEINAMEN . . . 32
SÄTZE EINER DATEI ÄNDERN . . . 32
DIE AKTUELLE DATEI . . . 33
LAGERHALTUNG . . . 33
Die Lager-Datei . . . 33
Die Lieferanten-Datei . . . 34
Die Bestell-Datei . . . 34
Anfragen . . . 35
Lager . . . 36
Verkaufsstatistik . . . 36
Hereinkommende Lieferungen . . . 36
Neubestellung . . . 37

KAPITEL 12
QL ARCHIVE ÜBERSICHT . . . 40
VARIABLEN . . . 40
SYNTAX . . . 40
AUSDRÜCKE . . . 40
Syntaktische Konventionen . . . 40
ARCHIVE DATEIEN . . . 41
Felder . . . 41
Datensätze . . . 42
Dateien . . . 42
Dateien eröffnen und schließen . . . 42
Logische Dateinamen . . . 42
PROZEDUREN . . . 42
DER PROGRAMM-EDITOR . . . 43
DER BILDSCHIRM-EDITOR . . . 43
DIE BEFEHLE . . . 44
ÄNDERN . . . 44
AKTIV . . . 44
ALLE . . . 44
ANF(ANG) . . . 44
ANLEGEN . . . 44

ANM(ERKUNG) 45
ANZEIGEN 45
AUSFÜHREN 45
AUSZUG 45
BANZEIGE 45
BEINGABE 45
BLADEN 45
BSICHERN 45
DRUCKEN 45
EDITIEREN 45
EDITOR 45
EINFÜGEN 46
EINGABE 46
ELIMINIEREN 46
ENDE 46
ENDEANLEGEN 46
ERSETZEN 46
ERÖFFNEN 47
ERWEITERN 47
EXPORT 47
FEHLER 47
FINDEN 47
FOLGESATZ 47
FORMATIEREN 47
IMPORT 47
LADEN 47
LEEREN 48
LISTEN 48
LÖSCHEN 48
LOKAL 48
MISCHEN 48
MODUS 48
NEU 48
NORMAL 48
ORDNEN 48
ORTEN 49
PAPIER 49
POSITION 49
PROTOKOLL 49
RESERVE 49
RÜCKKEHR 49
RÜCKSETZEN 49
SCHIRM 50
SCHLIESSEN 50
SCHRIFT 50
SETZEN 50
SICHERN 50
SICHTEN 50
SOLANGE 50
SPULEN 50
STOP 51
SUCHEN 51
VERLASSEN 51
VERZ 51
WÄHLEN 51
WEITER 51
WENN 51
ZEIGEN 51
ZURÜCK 52
FUNKTIONEN 52
ABS 52
ALLG 52
ANZAHL 52
BOG 52
CODE 52
DATEI ENDE 52
DATUM 52
DEZ 52
EXP 53
FEHLERNR 53
FELDNAME 53
FELDTYP 53
FELDWERT 53
FELDZAHL 53
GANZZAHL 53
GEFUNDEN 53
GRAD 53
GROSS 53
KETTE 53
KLEIN 53
KOS 53
LÄNGE 54
LN 54
MONAT 54
NUM 54
NUMWERT 54
PI 54
QWURZEL 54
SATZNR 54
SIN 54
SPEICHER 54
TAGE 54
TAN 54
TASTE 54
TEILKETTE 54
TIPPEN 55
VORZ 55
WERT 55
WIEDERH 55
WTN 55
ZCHN 55
ZEIT 55
FEHLER 55
Anmerkungen 56

sinclair

QL

QL Easel

WICHTIG

BEVOR SIE DIE APPLIKATIONEN BENUTZEN, LESEN SIE BITTE DIE NACHSTEHENDE BESCHREIBUNG, UM ARBEITSKOPIEN ZU ERSTELLEN.

ARBEITSKOPIEN

Fertigen Sie bitte immer mindestens eine Arbeitskopie an, bevor Sie eines der vier QL Programme benutzen. Arbeiten Sie dann immer nur mit der Arbeitskopie. Bewahren Sie die Originalkassetten mit den Programmen sicher auf. Benutzen Sie die Originalkassetten nur, um bei Bedarf Kopien anzufertigen. So schützen Sie sich vor dem Verlust Ihrer Programme.

Bei allen magnetischen Speichermedien, also auch bei Microdrive-Kassetten, kann bei Gebrauch Datenverlust entstehen. Daher sollten Sie alle Programme und Daten, die Sie noch benötigen, mindestens zweimal speichern. Fertigen Sie sich deshalb immer Sicherungskopien an. So schützen Sie sich vor Datenverlust. Immer, wenn Sie einer Kassette Daten oder Programme hinzufügen, sollten Sie das auch bei der Sicherungskopie tun. Wenn Sie das versäumen, laufen Sie Gefahr, alles, was Sie seit der letzten Datensicherung hinzugefügt haben, zu verlieren. Hinweise zur Anfertigung von Sicherungskopien finden Sie im Abschnitt *Information.*

by Dick de Grandis-Harrison *(Psion Limited)*

KAPITEL 1

WAS IST QL EASEL?

QL Easel ist ein Geschäftsgrafikprogramm, welches vollkommen *interaktiv* arbeitet. Sofort nach Einlegen der Programmkassette entstehen vor Ihren Augen grafische Darstellungen. Die Werte geben Sie entweder von Hand ein oder Sie importieren sie beispielsweise aus QL Abacus. Das Erstellen von Wertetabellen können Sie getrost vergessen. Easel besorgt das automatisch, ganz ohne Ihr Zutun.

Ob *Text* oder Daten, die Eingabe ist völlig unkompliziert, und was einmal in einer Grafik drinsteht, kann ohne weiteres editiert und solange umherbewegt werden, bis Sie mit der grafischen Präsentation vollauf zufrieden sind.

Easel ist stufenförmig aufgebaut und zeichnet sich durch seine *Pyramidenstruktur* aus. Die Spitze, die gleich von Anfang an zugänglich ist, gestattet die Durchführung der am häufigsten gebrauchten Operationen, etwa die Eingabe von Daten und Text. Die gesamte Palette der Variationsmöglichkeiten und die Leistungsfähigkeit von Easel offenbaren sich nach und nach mit zunehmender Vertrautheit mit dem Programm und tieferem Einstieg in die Pyramide.

Trotz seiner enormen Leistungsfähigkeit bleibt Easel auf allen Stufen einfach in der Handhabung. Sie brauchen sich nicht Mengen von Zahlen und Befehlen zu merken, da Easel Sie anhand von wohldurchdachten "prompts" (Systemmeldungen) durch alle Vorgänge führt und Sie laufend über die möglichen Schritte informiert. Außerdem gibt es die *Grafik nach Vorlage*-Funktion, mit der Sie, angefangen von einem einzelnen Strich oder Balken bis zu einer vollständigen Grafik, beliebige Elemente herausgreifen und entwerfen können – anhand einer Auswahl von Bildern. Auf diese Weise haben Sie völlige Gestaltungsfreiheit und Kontrolle über das endgültige Aussehen Ihrer Diagramme.

Falls Sie irgendwann nicht sicher sind, wie es weitergeht, holen Sie sich mit der Funktionstaste **F1** Hilfe. Vergessen Sie auch nicht, daß irrtümlich angeforderte, noch nicht ausgeführte Operationen mit **ESC** rückgängig gemacht werden können.

KAPITEL 2
FANGEN SIE AN

QL EASEL LADEN

Laden Sie QL Easel gemäß der Anleitung in der Einführung zu den QL-Programmen. Nach abgeschlossener Ladeprozedur meldet sich das System mit dieser Anzeige:

```
LADEN DER QL EASEL
Hochleistungsgrafik
Version x.y
Copyright ©1984 PSION Ltd.
Alle Rechte vorbehalten
```

wobei x.y die Versions-Nummer ist, z.B. 2.23.

Easel holt sich von Zeit zu Zeit weitere Daten von der Microdrive-Kassette. **Aus diesem Grund darf die Kassette nicht aus dem Microdrive 1 entfernt werden, bevor Sie Ihre Arbeit mit Easel beenden und auf SuperBASIC umsteigen.**

DIE BILDSCHIRMMASKE

Nach dem Ladevorgang sollte die Bildschirmmaske auf Ihrem Sichtgerät der Abbildung 2.1 entsprechen. Sie ist in drei Bereiche unterteilt: den Statusbereich, den Anzeigebereich und den Steuerbereich.

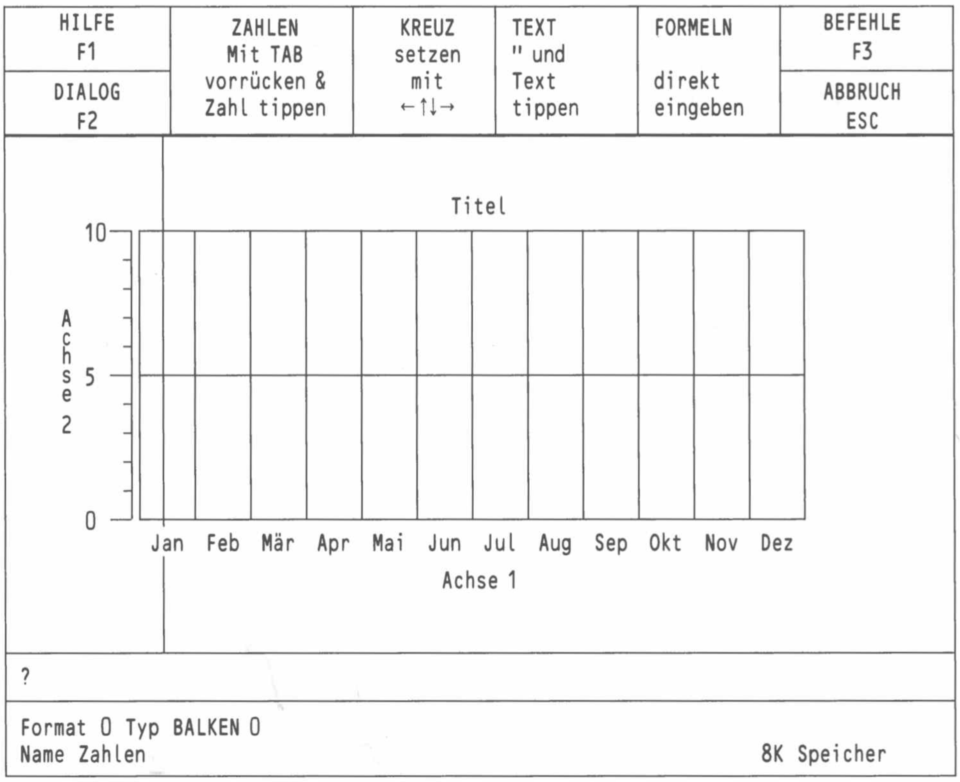


Abbildung 2.1 Die Hauptanzeige

Der Statusbereich

Das *Format* zeigt die Darstellungsweise der Eingabewerte an. Easel verfügt über acht verschiedene vordefinierte Anzeigeformate (numeriert von 0 bis 7), die eine Auswahl von Strich-, Balken- und Kreisdiagrammen ermöglichen. Die Standardeinstellung 0 erzeugt Balkendiagramme (Format 0).

Ferner wird im Statusbereich der Name der Daten- oder Zahlenserie Ihrer Grafik angezeigt. Zu jeder Grafik gehört eine solche Serie von Zahlen, die mit einem eigenen Namen versehen ist. Wenn Sie eine Eingabe vornehmen, betrifft die Änderung immer die aktuelle Zahlenserie.

Die dritte Rubrik ist der aktivierte *Grafiktyp*. Easel kann Zahlenserien in einer von drei Typen darstellen: als Strich-, als Balken- (Säulen-) oder als Tortendiagramm (auch Kreis- oder Kuchen-Diagramm genannt). Standardmäßig sieht Easel ein Balkendiagramm, Nummer 0, vor; insgesamt haben Sie die Wahl unter 16 verschiedenen Balkengrafiken.

Schließlich dient der Statusbereich zur Anzeige des frei verfügbaren Speicherplatzes und gegebenenfalls zur Ausgabe von *Fehlermeldungen*.

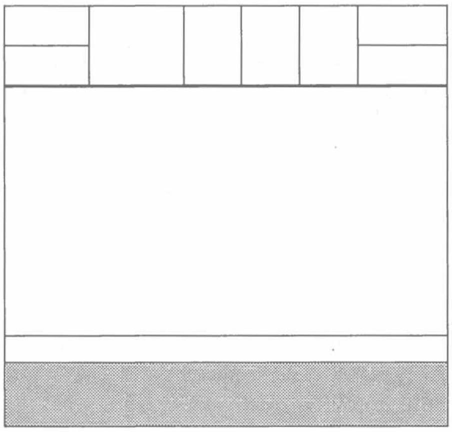


Abbildung 2.2 Der Statusbereich

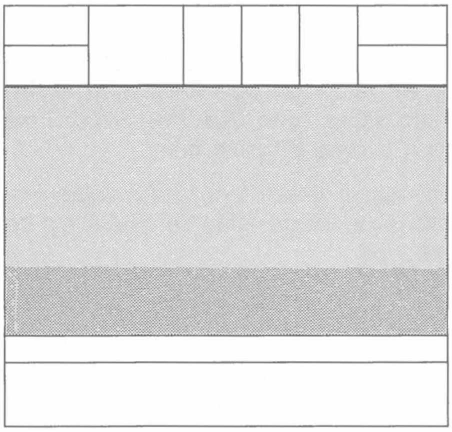


Abbildung 2.3 Der Anzeigebereich

Der Anzeigebereich

Hier werden alle Easel-Grafiken angezeigt.

Direkt nach dem Laden von Easel sehen Sie im Anzeigebereich ein leeres Koordinatengitter als Hintergrund für ein Balkendiagramm. Die horizontalen Linien entsprechen den Werten entlang der vertikalen Achse ("Achse 2"), und die vertikalen Linien bilden eine Unterteilung in *Felder* (oder Zellen). Jedes dieser Felder markiert die Position, an der jeweils ein Wert aus einer Zahlenserie eingezeichnet werden kann.

Zu jedem Feld gehört eine *Bezeichnung* auf der horizontalen Achse ("Achse 1"). Easel sieht automatisch die Namen "Jan", "Feb" ... bis "Dez" als Beschriftung vor, doch lassen sie sich beliebig ändern.

Stellen Sie sich eine Zahlenserie (ein Zahlenbild) als eine Reihe von Feldern vor, jedes mit einem Wert, der einen Bestandteil der Grafik ausmacht.

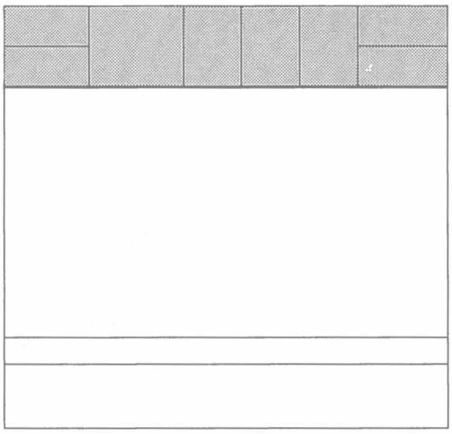


Abbildung 2.4 Der Steuerbereich

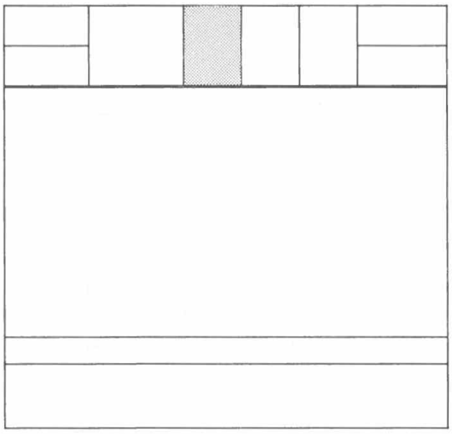


Abbildung 2.5 Das Fadenkreuz

Der Steuerbereich

Im Steuerbereich erkennen Sie die allgemeinen Funktionen, Hilfe (**F1**), Ein- und Ausschalten der Dialogmeldungen (**F2**), Aufrufen eines Befehls (**F3**) und Abbrechen einer irrtümlich angeforderten Operation (**ESC**). Darüber hinaus gibt es vier Easel-spezifische Optionen, nämlich:

Fadenkreuz bewegen,
Zahl eingeben,
Text eingeben,
Formel eingeben.

DAS FADENKREUZ

Drücken Sie die Rechtspfeiltaste, ohne sie gleich loszulassen. Dabei bewegt sich der *vertikale Strich* von links nach rechts über den Bildschirm. Die Bewegung in der anderen Richtung erfolgt mit dem Linkspfeil.

Die Auf- und Abwärtstasten bewegen den *horizontalen Strich*.

Jeder beliebige Punkt im Diagramm kann markiert werden, indem der Schnittpunkt (der Ursprung) des Fadenkreuzes dorthin verlegt wird.

Außerdem kennzeichnet der vertikale Strich die Position im Diagramm, an welcher die nächste Zahl eingezeichnet wird.

Zum Sichtbarmachen des Fadenkreuzes kann eine beliebige Cursortaste gedrückt werden: der Links– oder Rechtspfeil für den vertikalen Strich, der Auf– oder Abwärtspfeil für den horizontalen. Beachten Sie, daß dies nur in der Hauptanzeige möglich ist, nicht jedoch in dem Befehlsmenü.

Durch kurzes Anschlagen und Freigeben der Pfeiltasten bewegt sich der Strich um einen Schritt in der entsprechenden Richtung; bei längerem Festhalten beschleunigt sich die Bewegung.

ZAHLEN

Geben Sie eine Zahl ein (gefolgt von **ENTER**). Sie wird sofort an der aktuellen Position des senkrechten Strichs in Balkenform dargestellt, während sich der Strich um ein Feld weiter nach rechts bewegt, bereit für die nächste Eingabe.

Jede Zahl, welche den Zahlenbereich der Skala entlang der Achse 2 übersteigt, bewirkt ein erneutes Zeichnen der Grafik deren Skalierung dann die Anzeige des neuen Wertes erlaubt.

Mit jedem Tastendruck auf **TAB** springt der vertikale Strich um ein Feld nach rechts. Gleichzeitiges Festhalten von **SHIFT** und Anschlagen von **TAB** versetzt ihn um ein Feld nach links. Die Position des senkrechten Strichs markiert immer das *aktuelle Feld* – das Feld, in dem die nächste Zahl angezeigt wird, die Sie eingeben.

Eine fehlerhafte Eingabe in der Grafik können Sie ganz einfach so korrigieren, daß Sie den vertikalen Strich in das betreffende Feld bringen und dort den richtigen Wert eingeben.

Wenn Sie den Fehler bemerken, bevor Sie die Eingabe mit **ENTER** abschließen, verbessern Sie ihn entweder mit dem Zeileneditor oder stornieren ihn durch Drücken von **ESC** und Eingabe der richtigen Zahl.

Der vertikale Strich markiert stets das Feld, in dem der nächste Wert erscheint, unabhängig davon, ob die Bewegung mit **TAB** oder mit den Pfeiltasten erfolgt.

TEXT

Easel akzeptiert auch Text als Eingabe, vorausgesetzt, das erste Zeichen ist ein einfaches oder doppeltes Anführungszeichen (' oder ").

Das Fadenkreuz wird in der Grafik angezeigt (falls es nicht bereits vorhanden ist), und der eingegebene Text erscheint im Schnittpunkt des Kreuzes – und natürlich auch in der Eingabezeile im Statusbereich. Schließen Sie die Eingabe mit **ENTER** ab.

Der Text kann ohne weiteres in eine andere Position verschoben werden. Zu diesem Zweck benutzen Sie die Pfeiltasten, welche das Fadenkreuz bewegen, wobei der Text einfach mitgezogen wird. Sobald er an der richtigen Stelle sitzt, drücken Sie **ENTER**, worauf das Fadenkreuz unsichtbar wird.

FORMELN

Formeln dienen entweder zum Anlegen einer neuen oder zum Verändern einer bereits vorliegenden Zahlenserie.

Easel interpretiert jede Tastatureingabe, deren erstes Zeichen keine Zahl und kein Anführungszeichen ist, als eine Formel. Beispielsweise können wir die aktuelle Zahlenserie ändern, die den Namen "Zahlen" hat, wie Sie dem Statusbereich entnehmen können.

`Zahlen = Zahlen + 2` ENTER

Die neue Grafik ist sehr änlich wie die ursprüngliche, nur daß jeder Wert um 2 erhöht worden ist. Die Rückkehr auf die alte Grafik wird mit einer zweiten Formel bewerkstelligt:

`Zahlen = Zahlen - 2` ENTER

Formeln beginnen immer mit dem Namen einer Zahlenserie, ob dies nun der Name einer bereits bestehenden Serie ist oder ein neuer. Wie dem auch sei, der Inhalt der genannten Datenserie ist definiert durch den Ausdruck auf der rechten Seite des Gleichheitszeichens in der Formel. Dabei muß bedacht werden, daß die Formel alle Werte des Zahlenbilds beeinflußt, nicht etwa nur einen.

DIE BEFEHLE

Die Befehle verschaffen Ihnen den Zugang zu den leistungsfähigeren Funktionen von Easel. Sie erreichen sie über die Funktionstaste **F3**, worauf im Steuerbereich das Befehlsmenü angezeigt wird.

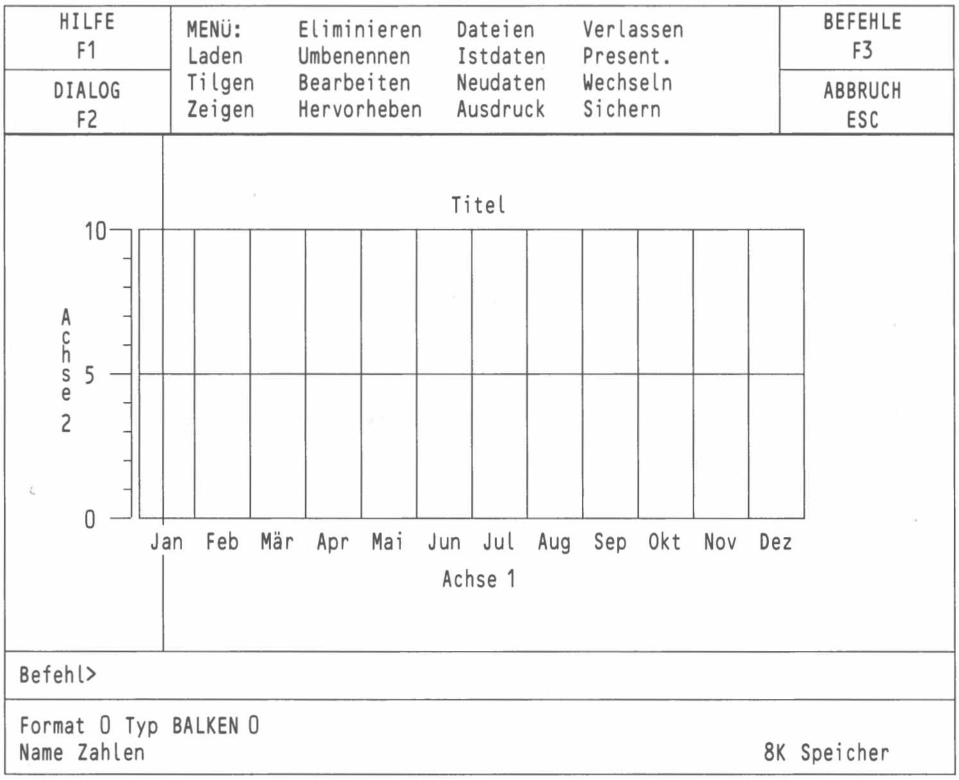


Abbildung 2.6 Das Easel-Befehlsmenü

Aus der Liste der verfügbaren Befehle können Sie durch Eingabe des entsprechenden Anfangsbuchstabens den gewünschten Befehl anwählen.

Der Befehl **Verlassen** z.B. dient zum Aussteigen aus Easel und bewirkt die Rückkehr in SuperBASIC. Aufgerufen wird er mit **F3** und Eingabe von **V**, worauf Ihnen Easel Gelegenheit gibt, es sich noch einmal mit **ESC** anders zu überlegen und in Easel zu bleiben (falls Sie den Befehl versehentlich gedrückt haben). Zum Verlassen von Easel drücken Sie **ENTER**.

Solange im Steuerbereich das Befehlsmenü angezeigt wird, ist die Eingabe in die Felder gesperrt, und ebenso die Verschiebung des Fadenkreuzes, es sei denn, dies ist eine befehlsinterne Option.

Nach Durchführung eines Befehls bleibt Easel im Befehlsmenü. Die Rückkehr zur Hauptanzeige erfolgt mit **ESC**.

WERTE LÖSCHEN

Werte können auch wieder gelöscht werden. Mit **TAB** (bzw. **SHIFT** und **TAB**) plazieren Sie den vertikalen Strich auf die Zahl, die gelöscht werden soll, und drücken dann **F4**. Falls Ihre Grafik mehr als eine Zahlenserie darstellt, löscht **F4** alle in dem betreffenden Feld angezeigten Werte. Nicht beeinträchtigt werden Zahlenserien, die nicht angezeigt sind. Nicht beschriftete Felder, deren Wert gelöscht wird, entfallen beim nächsten Neuzeichnen der Grafik.

Easel löscht nur solche Felder, die weder eine Beschriftung noch einen Wert haben. Voraussetzung zum Löschen eines Feldes ist also, daß vorher sein Inhalt und gegebenenfalls auch seine Beschriftung gelöscht wird. Ein einmal gelöschtes Feld wird beim nächsten **Zeigen**-Befehl nicht mehr angezeigt.

WERTE EINFÜGEN

Ein neuer Wert kann rechts von dem durch den vertikalen Strich markierten eingefügt werden. Durch Drücken von **F5** öffnet sich die Grafik zür Eingabe eines neuen Wertes. Das auf diese Weise gebildete Feld wird nicht automatisch mit einer Beschriftung versehen, doch können Sie das selbstverständlich nachholen.

Bei Tortendiagrammen geht das Einfügen und Löschen von Werten ein wenig anders vor sich. Für eine ausführliche Beschreibung verweisen wir auf Kapitel 9.

KAPITEL 3

GESTALTUNG VON BALKEN

Dieses Kapitel zeigt Ihnen verschiedene Variationsmöglichkeiten für die optische Gestaltung Ihrer Balkendiagramme.

Alle verfügbaren Optionen funktionieren nach demselben Prinzip. Wir beschränken uns hier auf die Beschreibung der Methode zum Ändern der Balken, die als Beispiel für die anderen Optionen gelten soll.

Wir gehen davon aus, daß Sie einige Zahlen eingegeben und ein Balkendiagramm vor sich auf dem Bildschirm haben.

EINE BALKENFORM WÄHLEN

Zur Auswahl einer anderen Balkenart rufen Sie den **Wechseln**-Befehl auf, den Sie über **F3** und **W** erreichen. Der Befehl verfügt über ein eigenes, reichhaltiges Menü – zum Modifizieren der Balken, Linien, des Koordinatengitters, usw. Mit **B** wählen Sie die Balken-Option.

Zum Wählen eines neuen Balkenstils stehen zwei Möglichkeiten offen: durch Eingabe einer Nummer oder anhand von Beispielen.

Wahl mit Nummer

Nach Aufrufen der Balken-Option zeigt die Eingabezeile den Text:

```
BEFEHL>Wechsel zu BALKEN ?
```

und Easel erwartet eine Nummer. Sie haben die Qual der Wahl unter 16 verschiedenen Balkenformen, numeriert von 0 bis 15, die Sie durch Eingabe der betreffenden Zahl, gefolgt von **ENTER** anfordern.

Diese Methode ist äußerst schnell, wenn Sie bereits wissen, was für einen Balkentyp Sie wünschen und welche Nummer er hat.

Wahl anhand von Beispielen

Wenn Sie hingegen die Nummer nicht kennen oder vorziehen, eine eigene Balkenform zu entwerfen, dann drücken Sie stattdessen **ENTER**. Probieren Sie diesen Weg einmal aus. Die erforderliche Eingabesequenz ist:

[F3] W B [ENTER]

(Das Drücken von **F3** entfällt, wenn Sie sich immer noch im Befehlsmenü befinden.) Im Anzeigebereich erscheint nun die ganze Palette der verfügbaren Balken, zusammen mit ihren Nummern.

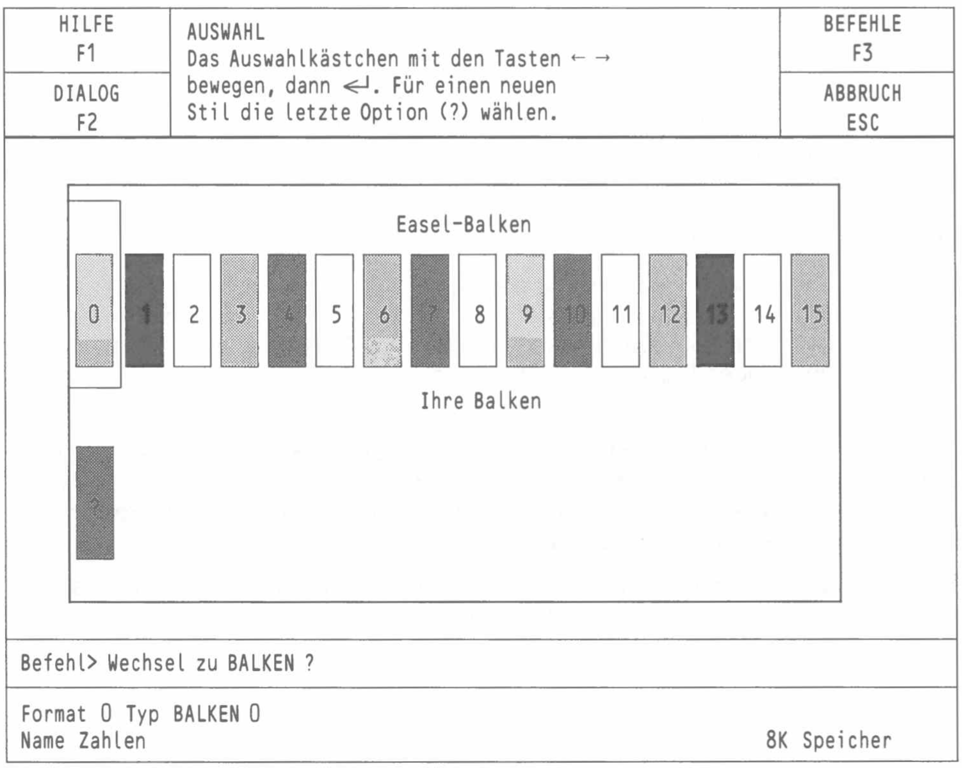


Abbildung 3.1 Eine Balkenform wählen

Der im Moment gültige Typ ist von einem hochstehenden Feld umrandet, eine Art Menü-Cursor, den Sie mit den Pfeiltasten nach rechts und nach links verschieben können, um den gewünschten Balken anzusteuern. Die Wahl bestätigen Sie mit **ENTER**.

EIGENENTWURF

Bei Verwendung der Auswahl anhand von Beispielen ist Ihnen sicher der Balken in der unteren Reihe aufgefallen, der anstelle einer Nummer ein Fragezeichen aufweist. Dieser Balken muß gewählt werden, wenn Sie vorhaben, einen eigenen Balkentyp zu entwerfen.

Verlagern Sie den Menü-Cursor auf diesen Balken und drücken Sie **ENTER**. Darauf werden Ihnen ein Musterbalken und verschiedene Optionen zur Gestaltung präsentiert.

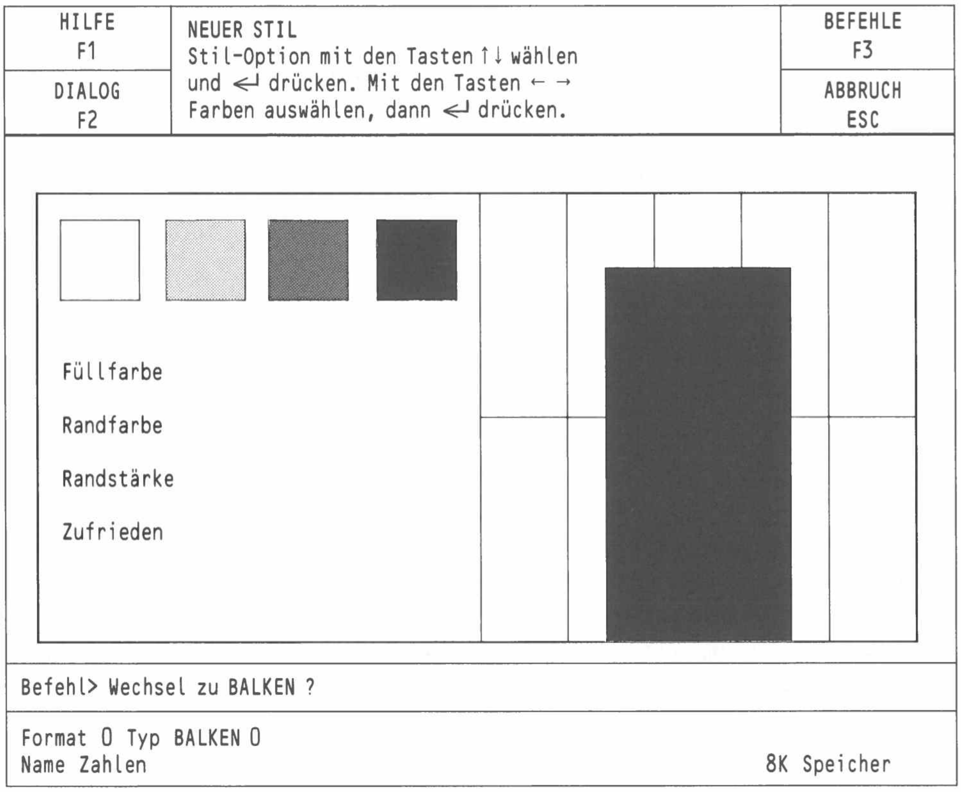


Abbildung 3.2 Selbst einen Balkentyp entwerfen

Balken-Farbe

Die erste farblich hervorgehobene, aktivierte Option ist die Farbe zum Ausfüllen (Schraffieren) des Balkens, welche aus der angezeigten Palette zu wählen ist. Akzeptieren der Option mit **ENTER** bzw. Aufsuchen einer anderen Auswahlzeile mit dem Auf– und Abwärtspfeil.

Zur Festlegung der Füllfarbe gibt es einen quadratischen Auswahlrahmen, welcher zu Anfang auf dem ersten Farbblock plaziert ist. Der Musterbalken ist mit dieser Farbe gefüllt. Steuern Sie das Quadrat mit Hilfe der Rechts– und Linkspfeiltasten und drücken Sie **ENTER**, sobald der Musterbalken in der gewünschten Farbe erscheint. Easel malt den Balken gegen den Hintergrund des aktuellen Koordinatenpapiers.

Randfarbe

Als nächstes wird die Option "Randfarbe" hervorgehoben, die Sie entweder durch Drücken von **ENTER** akzeptieren oder überspringen. Die Farbauswahl verläuft hier genau wie bei der Füllfarbe. (Bei einer Randstärkeneinstellung von 0 ist natürlich die Randfarbe nicht erkennbar.) Sobald Sie mit dem Resultat zufrieden sind, bestätigen Sie dies durch Drücken von **ENTER**.

Randstärke

Die dritte Option betrifft die Randstärke, welche Sie anhand einer Zahl zwischen 0-100 (Prozentwert im Verhältnis zur halben Balkenbreite) bestimmen.

Schließlich kommt die Frage, ob Sie mit Ihrem Eigenentwurf zufrieden sind. Wenn ja, drücken Sie **ENTER**, wodurch der neue Balkentyp in die bestehende Liste aufgenommen und automatisch zur Darstellung der aktuellen Zahlenserie verwendet wird. Andernfalls kehren Sie mit Hilfe des Auf– und Abwärtspfeils zu den Optionen zurück, die Sie verändern möchten, und versuchen es mit einer anderen Kombination. Sie haben vor der endgültigen Eingabe des neuen Typs auch jederzeit die Wahl, aus dem Befehl mit **ESC** auszusteigen, ohne einen neuen Balkentyp entworfen zu haben.

KAPITEL 4
VERWENDUNG VON TEXT

Neu eingegebener oder überarbeiteter Text wird immer in der Farbe und Ausrichtung (horizontal, vertikal) angezeigt, die Sie zuletzt mit der Text-Option im Menü des **Wechseln**-Befehls vorgesehen haben.

Easel unterscheidet drei grundsätzliche Textarten:

Gewöhnlicher Text (einschließlich des Titels)
Achsenbeschriftung
Feldnamen (Bezeichnungen).

GEWÖHNLICHER TEXT

Gewöhnlicher Text, d.h. alle alphabetischen Zeichenketten mit Ausnahme der Achsenbeschriftungen und der Feldnamen, wird sozusagen auf die Grafik aufgeklebt und bleibt solange dort, bis er gelöscht wird, unabhängig von etwaigen anderen Änderungen.

Der **Bearbeiten**-Befehl verfügt über Optionen zum Ändern dieser drei Textarten und über eine vierte für die Legende. Diese letztere ist nur dann von Bedeutung, wenn mit mehr als einer Zahlenserie gleichzeitig gearbeitet wird. Eine ausführliche Beschreibung folgt im nächsten Kapitel.

Im **Bearbeiten**-Menü **T** für Text drücken. Dann bewegen Sie das Fadenkreuz so, daß die Schnittstelle in die Nähe des zu ändernden Textes zu liegen kommt. Die exakte Stelle brauchen Sie nicht anzusteuern: bei Drücken von **ENTER** positioniert sich das Fadenkreuz automatisch auf das nächstliegende Textstück. Gleichzeitig wird der Text auch in der Eingabezeile angezeigt und dort zur Bearbeitung freigegeben.

Zum Löschen drücken Sie **F4**, zum Modifizieren verwenden Sie den Zeileneditor. Löschen bewirkt automatische Beendigung des Befehls.

Nach Ausführung etwaiger Änderungen bestätigen Sie die endgültige Fassung mit **ENTER**. Easel gibt Ihnen dann die Gelegenheit, den Text anhand der Pfeiltasten zu verschieben. Mit **ENTER** bestätigen Sie die richtige Position.

Für Easel ist der Titel einer Grafik dasselbe wie gewöhnlicher Text, mit dem Unterschied, daß beim Laden von Easel gleich die Standardüberschrift "Titel" zentriert über die Grafik gesetzt wird.

ACHSEN-NAMEN

Achsenbeschriftungen gibt es sinngemäß nur bei den Balken– und Liniengrafiken, nicht jedoch bei den Tortendiagrammen.

Wählen Sie aus dem internen Menü des Befehls **Bearbeiten A** zum Ändern einer der beiden Achsenbeschriftungen: **V** ist die vertikale, **H** die horizontale Achse. Auch hier kann der Text überarbeitet, gelöscht oder verschoben werden, analog zur Textoption. Easel schreibt den Text in der aktuellen Schrift– und Papierfarbe.

FELDNAMEN

Die Koordinatenfelder sind standardmäßig mit den Abkürzungen für die zwölf Monate versehen, wobei die Beschriftung bei Balken– und Liniendiagrammen entlang der horizontalen Achse läuft und bei Tortendiagrammen die einzelnen Kreissegmente markiert.

Für eine Änderung der Feldnamen dient die Option **B** "Bezeichnungen" innerhalb des **Bearbeiten**-Untermenüs. Dabei verschiebt sich das Fadenkreuz automatisch auf den nächstliegenden Namen, der auch in der Eingabezeile erscheint. Feldnamen (Bezeichnungen) dürfen bis maximal zehn Zeichen lang sein, obwohl im allgemeinen nur die ersten paar Buchstaben sichtbar sind. Die Überarbeitung erfolgt mit dem Zeileneditor, zum Löschen dient die Funktionstaste **F4**. **ENTER** schließt den Bearbeitungsvorgang ab. Sobald Sie eine Bezeichnung editieren, wechselt die ursprüngliche Farbe auf die aktuelle Textfarbe. Feldnamen können nicht verschoben werden.

TEXTFARBE UND RICHTUNG

Die Text– und die Hintergrundfarbe ändern Sie mit der Text-Option **(T)** von **Wechseln**. Außerdem können Sie die Richtung (vertikal oder horizontal) bestimmen.

Die neue Textfarbe und –richtung gilt für alle Texteingaben, die von dem Augenblick an erfolgen, und auch für bereits bestehenden Text, der überarbeitet wird.

Um die Textfarbe zu ändern, ist es am einfachsten, die entsprechende Option zu wählen und danach den Befehl **Bearbeiten** für bestehenden Text zu verwenden, ohne dabei irgendwelche Änderungen in der Formulierung oder der Position vorzunehmen.

Wählen Sie die Text-Option im **Wechseln**-Befehl. Hier offeriert Easel eine Liste von Textstil-Optionen – ganz ähnlich wie beim Entwurf eines eigenen Balkentyps. Auch hier bewegen Sie sich mit Hilfe der Auf– und Abwärtspfeiltasten auf die gewünschten Auswahlzeilen, wobei jeweils die farblich abgesetzte aktiviert und mit **ENTER** gewählt wird.

Textfarbe

Bei der ersten Option geht es um die Textfarbe. Mit dem Rechts– und dem Linkspfeil verschieben Sie den "Rahmen" auf die gewünschte Farbe und bestätigen diese Wahl mit **ENTER**, worauf automatisch die folgende Auswahlzeile hervorgehoben und zur Änderung freigegeben wird. Zum Anwählen **ENTER** drücken.

Papierfarbe – Texthintergrund

Mit der zweiten Option bestimmen Sie die Hintergrundfarbe. Auch hier bewegen Sie den "Rahmen" mit dem Rechts– und Linkspfeil entlang der Farbpalette und drücken zur Bestätigung und zum Übergang auf die nächste Option **ENTER**.

Die dritte Option wählt einen durchsichtigen Hintergrund für den Text. Bei Wahl dieser Möglichkeit ignoriert Easel die eingestellte Papierfarbe, so daß der Diagrammhintergrund um den Text herum sichtbar bleibt. Jedesmal, wenn Sie diese Option eingeben, wechselt Easel zwischen dem durchsichtigen und dem ausgewählten Farbhintergrund.

Textrichtung

Die vierte Option erlaubt Umschaltung zwischen horizontaler und vertikaler Textdarstellung.

Schließlich ist es an Ihnen zu entscheiden, ob die gewählte Textanzeige Ihren Vorstellungen entspricht. Wenn ja, bestätigen Sie den Textstil mit **ENTER**, worauf Sie wieder ins Befehlsmenü zurückkehren. Andernfalls springen Sie mit den Auf– und Abwärtspfeilen nochmals auf die Optionen, die geändert werden sollen.

KAPITEL 5
MEHRERE ZAHLENBILDER

Bisher haben wir uns auf die Eingabe und Darstellung einer einzigen Zahlenserie beschränkt. Nun gibt es jedoch Anwendungen, wo es sich als vorteilhaft erweist, zwei oder mehr Zahlenserien in einem einzigen Diagramm zu kombinieren, beispielsweise für einen Vergleich zwischen den Umsatzzahlen des Vorjahres und des laufenden Jahres. Wie Sie es anstellen, innerhalb einer Grafik mehrere Zahlenserien zu erzeugen, zu bearbeiten und anzuzeigen, lernen Sie im Verlauf dieses Kapitels.

DIE AKTUELLEN ZAHLEN

Ganz gleich, wieviele Zahlenserien zu einer Grafik gehören, es kann gleichzeitig immer nur eine bearbeitet werden, die sogenannte *aktuelle Zahlenserie*, deren Name im Statusbereich angezeigt wird. Beim Starten von Easel arbeiten Sie zunächst mit einer Zahlenserie, welche standardmäßig den Namen "Zahlen" hat. Die aktuelle Serie wird immer in der Bildschirmmaske angezeigt.

DER UMBENENNEN-BEFEHL

Nehmen wir an, Sie haben nach Laden von Easel eine Zahlenserie eingegeben und wollen den Standardnamen "Zahlen" auf "Verkauf" ändern. Zu diesem Zweck gibt es den Befehl **Umbenennen**, den Sie über **F3** und **U** erreichen. Easel bittet um Eingabe (bzw. Bestätigung) des alten Namens, gefolgt von **ENTER**. Danach schreiben Sie den neuen Namen, den Sie ebenfalls mit **ENTER** abschließen. Die Eingabesequenz für die vorgeschlagene Änderung von "Zahlen" auf "Verkauf" lautet:

[F3] `U Zahlen` [ENTER] `Verkauf` [ENTER]

Die auf diese Weise umbenannte Serie wird zum aktuellen Zahlenbild.

DER NEUDATEN-BEFEHL

Es gibt zwei Methoden, um nacheinander Datenserien einzugeben: entweder mit dem Befehl **Neudaten** oder mit Hilfe einer Formel. Diese beiden Methoden werden im folgenden beschrieben.

Angenommen, Sie haben gemäß obiger Anleitung ein Zahlenbild namens "Verkauf" zusammengestellt, welches die monatlichen Verkaufszahlen enthält. Diesen Zahlen sollen innerhalb der gleichen Grafik die monatlichen Kosten gegenübergestellt werden. Rufen Sie mit **F3** und **N** den Befehl **Neudaten** auf und tragen Sie für die neue Zahlenserie einen Namen ein. Die Eingabe wie immer mit **ENTER** abschließen.

Die ganze Eingabesequenz zum Anlegen einer neuen Zahlenserie unter dem Namen "Kosten" lautet folglich:

[F3] `N Kosten` [ENTER]

Easel legt Ihnen sofort ein neues, leeres Koordinatenpapier vor (vorausgesetzt, das Programm ist auf Balken– oder Liniendiagramm eingestellt), mit dem vertikalen Strich in der ersten Spalte. Ein Blick auf den Statusbereich informiert darüber, daß die neue Zahlenserie mit dem Namen "Kosten" nunmehr die aktuelle Serie ist. Alle jetzt eingegebenen Zahlen werden sofort in der Grafik angezeigt.

Eine dritte Zahlenserie kann genau gleich erstellt werden, ebenfalls unter Zuhilfenahme von **Neudaten** und unter Verwendung eines neuen Namens. Grundsätzlich können Sie beliebig viele Zahlenserien eingeben; die einzige natürliche Beschränkung ist der verfügbare Speicherplatz.

VERWENDUNG EINER FORMEL

Es kann vorkommen, daß Sie eine neue Zahlenserie eingeben wollen, die in einer systematischen Weise mit bereits vorliegenden Serien verknüpft ist.

Beispielsweise haben Sie alle Eintragungen für "Verkauf" und "Kosten" vorgenommen und wollen jetzt die daraus resultierenden Gewinne anzeigen. In einem solchen Fall genügt es, mit einer Formel das Verhältnis der neuen zu den bestehenden Werten zu definieren, etwa:

```
Gewinn = Verkauf - Kosten
```

Damit wird eine neue Zahlenserie mit dem Namen "Gewinn" erstellt, wobei jeder Wert die Differenz zwischen den betreffenden "Verkauf"– und "Kosten"-Werten ist. "Gewinn" ist in diesem Moment die aktuelle Zahlenserie, d.h. die in der Grafik dargestellte.

Formeln müssen sich nicht notwendigerweise aus Verweisen auf bestehende Zahlenserien zusammensetzen. Genausogut denkbar sind Formeln von der Art:

```
Kurve = 10 * sin(Feld/2)
```

Mit dieser Formel wird eine neue Zahlenserie mit dem Namen "Kurve" erstellt und angezeigt, deren Werte anhand der **sin** () Funktion berechnet werden. Außerdem taucht in der Formel die Bezeichnung "Feld" auf, welche die Feldnummer, ausgehend vom Wert 1 an der linken Seite der Grafik, angibt. Am besten sehen Sie die Wirkung durch Eingabe von:

```
a = Feld
```

Die Verwendung von "Feld" im Rahmen einer Formel führt dazu, daß die Anzahl der Werte in der aktuellen Zahlenserie identisch wird mit der Anzahl der im Diagramm enthaltenen Spalten.

Daneben gibt es noch ein anderes reserviertes Wort in Easel, "Feldmax". Sein Wert ist gleich der Anzahl der aktuell am Bildschirm gezeigten Felder. "Feldmax" dient u.a. zur Anpassung der horizontalen Achsenskala in einer Formel. So zeichnet z.B. die Formel

```
Kurve = sin(2*pi()*(Feld - 1)/(feldmax - 1))
```

einen kompletten Sinuskurvenzyklus, unabhängig von der Anzahl der am Bildschirm sichtbaren Felder.

DER ISTDATEN-BEFEHL

Bei Verwendung von **Neudaten** wird das neue Zahlenbild zum aktuellen. Eingaben oder Modifikationen können nur an diesem aktuellen Zahlenbild vorgenommen werden. Wenn es sich als notwendig erweist, Änderungen an einem anderen, bereits bestehenden Zahlenbild auszuführen, brauchen Sie einen anderen Befehl, **Istdaten**. Nach Aufrufen dieses Befehls mit **F3** und **I** werden Sie aufgefordert, den Namen des betreffenden Zahlenbildes einzugeben, womit Sie dieses zur aktuellen Zahlenserie machen.

Nehmen wir an, Sie haben drei Zahlenbilder mit den Namen "Verkauf", "Kosten" und "Gewinn", wobei "Gewinn" die aktuelle Zahlenserie ist. Um die Zahlen in der Serie "Kosten" zu überarbeiten, rufen Sie diese mit **Istdaten** auf und nehmen beliebige Änderungen vor.

Dabei ist zu beachten, daß sich die Modifikationen bei den "Kosten" nicht automatisch auf die "Gewinn"-Grafik auswirken. (Das ist eine Aufgabe für QL Abacus.)

DATEN ZEIGEN

Der Befehl **Zeigen** erlaubt Ihnen die gleichzeitige Darstellung aller Zahlenserien in einem einzigen Diagramm.

Rufen Sie ihn mit **F3** und **Z** auf. Easel macht den Vorschlag, alle Zahlenbilder anzuzeigen, was Sie durch Drücken von **ENTER** bestätigen können. Auch dem nächsten Vorschlag, dem Anzeigeformat 0, können Sie mit **ENTER** zustimmen, worauf die ganze Grafik sofort am Bildschirm erscheint, mit allen definierten Daten und einem *Legendenfeld*, dem der Name der einzelnen Zahlenserienbilder und die Anzeigeart zu entnehmen ist. (Diese Legende entfällt, wenn die Grafik nur ein Zahlenbild enthält.)

Wenn Sie eine große Menge von Zahlenserien definiert haben, darf es Sie nicht überraschen, daß Ihre Grafik überfüllt und nicht unbedingt aussagekräftig wirkt. Denken Sie an den Wald, den man vor lauter Bäumen nicht mehr sieht... Wirkungsvolle Grafiken erzielen Sie, wenn Sie nur wenige Zahlenserien kombinieren. Das heißt jedoch keineswegs, daß Sie sich bei der *Definition* von Zahlenserien zurückhalten sollen, sondern es geht darum, bei Verwendung des Befehls **Zeigen** die richtige Kombination zu wählen.

Das tun Sie ganz einfach durch Ablehnen des Vorschlags "Alle Zahlenbilder". Statt **ENTER** zu drücken, geben Sie die Liste der gewünschten Zahlenbilder, getrennt durch Kommas, ein. Erst dann drücken Sie **ENTER**.

Es steht Ihnen auch frei, ein anderes als das standardmäßig vorgesehene Anzeigeformat (0) zu wählen, indem Sie dort nicht einfach **ENTER** drücken, sondern eine Zahl zwischen 0 und 16 eingeben. Easel verfügt über acht vordefinierte Formate für die Anzeige verschiedener Arten von Balken-, Linien- und Tortendiagrammen. Durch Eingabe eines Fragezeichens holen Sie sich eine Aufstellung der möglichen Formate auf den Bildschirm. Lassen Sie sich mit **Zeigen** drei oder vier Zahlenbilder in mehreren Formaten anzeigen.

DIE LEGENDE

Eine der Optionen im internen Menü des Befehls **Bearbeiten** dient zum Verschieben des Legendenkästchens. Nach Drücken von **L** wird die Legende unsichtbar, und es bleibt lediglich das leere Kästchen, welches Sie mit **F4** löschen oder mit Hilfe der Pfeiltasten im Diagramm verschieben können. Sie können mit dem Kästchen beliebig umherfahren und die vorgesehene Position mit **ENTER** markieren, worauf das Diagramm sofort neu gezeichnet wird – mit der Legende an der neuen Stelle.

Falls Sie eine Legende wieder herstellen wollen, die Sie bereits gelöscht haben, verwenden Sie wieder den Befehl **Bearbeiten** mit der Legenden-Option, worauf der Rahmen des Kästchens angezeigt wird. Bewegen Sie es an die gewünschte Position, und drücken Sie **ENTER**, worauf das Diagramm neugezeichnet und die Legende wieder angezeigt wird.

Die einzige Änderung, die Sie am Inhalt des Legendenkästchens durchführen können, ist die Textfarbe. Sie bestimmt sich aus der zuletzt gewählten Farbe im Befehl **Wechseln**. Es versteht sich von selbst, daß die in der Legende verwendeten Symbole immer mit den in der Grafik angezeigten übereinstimmen.

KAPITEL 6 GRAFIK-FORMATE

DAS FORMAT ÄNDERN

Easel verfügt über eine Auswahl von acht *Anzeigeformaten* (numeriert von 0 bis 7). Bei jedem Aufruf des **Zeigen**-Befehls ist anhand einer Zahl das gewünschte Format zu spezifizieren. Es bestimmt nicht nur die Hintergrund– und die Textfarbe, sondern auch, welcher Anzeigestil zur Anwendung kommt.

Zur Festlegung des Formats kann auch die Form-Option des **Wechseln**-Befehls benutzt werden. In der Eingabezeile erscheint die Anfrage

```
BEFEHL> Wechsel zu Format ?
```

die Sie mit einer Nummer zwischen 1 und 7 beantworten können (gefolgt von **ENTER**). Bloßes Drücken von **ENTER** bewirkt die Anzeige der 8 Formate und die Anforderung um Eingabe der Format-Nummer.

FORMATE NEU DEFINIEREN

Es ist ohne weiteres möglich, eines oder mehrere der acht von Easel vorgesehenen Formate nach eigenem Geschmack neu zu definieren.

Dabei haben Sie eine ungefähre Vorstellung, wie Ihre Grafik aussehen soll. Wählen Sie das Standardformat, was dem am nächsten kommt, und ändern Sie es so, daß es Ihren Anforderungen entspricht.

Zum Ändern der Feldnamen und der Achsenbeschriftungen verwenden Sie den **Bearbeiten**-Befehl, zur Modifikation des Textstils und der Balkenarten den **Wechseln**-Befehl.

Für eine Liniengrafik bietet sich Format 3 an bzw. die Linien–Option im internen Menü des **Wechseln**-Befehls. Kreisdiagramme beschreiben wir in Kapitel 8.

Koordinatenpapier

Die Option Koordinatenpapier im **Wechseln**-Befehl gestattet Ihnen die Auswahl unter verschiedenen Papier– und Koordinatengitter–Farben. Standardmäßig sind 7 verschiedene Stile vorgesehen, aber auch hier können Sie Ihre eigenen Vorstellungen verwirklichen. Zum Entwerfen Ihres Koordinatenpapiers gehen Sie genau so vor wie in der Anleitung zur Definition eines Balkens.

Achse

Die Achsen-Option des **Wechseln**-Befehls präsentiert Ihnen 10 verschiedene Stile und die Möglichkeit, eine eigene Achse zu definieren. Das Vorgehen ist analog zu der unter "Balken" beschriebenen Methode, die auch für das Koordinatenpapier gilt.

Beim Achsen-Entwurf wählen Sie die Farben für die Achsen, bestimmen, ob die Achsen als Linien gezogen werden sollen oder nicht, und entscheiden über die Farbe der Skalenbeschriftung entlang der vertikalen Achse und die Achsenbegrenzungen.

Im allgemeinen wählt Easel die Begrenzungen, d.h. den Skalenbereich so, daß alle Werte aus Ihrer Zahlenserie erfaßt werden können. Es gibt drei Möglichkeiten zum Ändern dieser Option.

A wählt automatische Begrenzung. Damit überlassen Sie es Easel, einen optimalen Skalenbereich im Hinblick auf die vorliegenden Grafikwerte zu bestimmen. Die Skala enthält nicht notwendigerweise einen Nullpunkt, wenn es sich z.B. durchwegs um hohe, positive Werte handelt.

N wählt automatische Begrenzung unter Einschluß des Nullpunkts. Das ist die Standardskalenform von Easel, wenn Sie nicht ausdrücklich etwas anderes vorsehen.

M wählt manuelle Begrenzung. Easel fordert die Eingabe der beiden Endpunkte, erst des unteren (gefolgt von **ENTER**), dann des oberen (gefolgt von **ENTER**).

Die von Ihnen gewählte Skala wird von Easel ersetzt, falls sie nicht den gesamten aktuellen Zahlenbereich abdeckt.

In allen Fällen werden die beiden Endpunkte so festgesetzt, daß sich auf der Skala eine sinnvolle Einteilung der Teilstriche ergibt. Beachten Sie, daß die rechts angezeigte Musterachse nicht unbedingt den exakten Zahlenbereich anzeigt, der Ihrer Grafik zugrundegelegt wird. Sie zeigt lediglich den Typ Achse, den Sie gewählt haben.

KAPITEL 7 LINIEN-DIAGRAMME

Sicher ist Ihnen beim Experimentieren mit den verschiedenen Anzeigeformaten bereits aufgefallen, daß neben den sog. Balken– oder Säulendiagrammen auch Darstellungen in Form von Linien– und Tortengrafiken vorgesehen sind. Mit Easel ist es möglich, die jeweils vorteilhafteste Präsentationstechnik für eine Zahlenserie zu wählen.

Format 3 verwendet Liniendiagramme zur Zahlendarstellung. Jeder Wert kann durch ein Symbol markiert und durch eine Linie beliebiger Farbe und Stärke mit den benachbarten verbunden werden. Auf Wunsch sind auch "gefüllte" Linien verfügbar, wobei der gesamte Bereich zwischen der Linie und der horizontalen Achse mit Farbe ausgemalt wird.

Solche gefüllten Linien werden z.B. verwendet zur Darstellung von "kritischen Werten", etwa der Schwelle, wo Kosten und Ertrag gleich sind.

Da Balken– und Liniendiagramme auf der gleichen Art von Koordinatenpapier dargestellt werden, lassen sie sich beliebig kombinieren. Titel, Achsenbeschriftungen, gewöhnlicher Text und das Legendenkästchen verhalten sich bei beiden Diagrammtypen genau gleich.

VERSCHIEDENE LINIENARTEN

Durch Aufrufen der Linien-Option im internen **Wechseln**-Menü verwandeln Sie eine Zahlenserie in ein Liniendiagramm. Als erstes machen Sie das dafür vorgesehene Zahlenbild zum aktuellen (z.B. mit **Istdaten**) und gelangen dann mit **F3**, **W** und **L** zu der Linien-Option.

Easel kennt 16 vordefinierte Linienstile und bittet Sie um Eingabe der entsprechenden Nummer. Wenn Sie Ihrer Sache sicher sind, geben Sie diese ein, gefolgt von **ENTER**; andernfalls drücken Sie einfach **ENTER**, worauf Easel Ihnen die 16 Stile zur Auswahl vorlegt. Bewegen Sie das Auswahlfeld mit Hilfe des Rechts– und Linkspfeils und treffen Sie Ihre Wahl durch Drücken von **ENTER**. Sofort wird das Diagramm in dem neuen Stil gezeichnet.

Wenn keine der vorgegebenen Stile Ihren Ansprüchen genügt, steht es Ihnen frei, eine eigene Linie zu entwerfen. Zu diesem Zweck steuern Sie die Linie mit dem Fragezeichen an.

Auch hier arbeiten Sie anhand verschiedener Optionen, und das Vorgehen ist gleich wie beim Entwurf eines Balkens, wie Sie es in Kapitel 3 gesehen haben. **ENTER** aktiviert jeweils die hervorgehobene Option; das Springen von Auswahlzeile zu Auswahlzeile erfolgt mit Hilfe des Auf– und Abwärtspfeils.

Linien-Farbe

Zur Auswahl der Farbe mit dem Rechts– und Linkspfeil in der Palette hin– und herfahren. Zur Bestätigung und Sprung auf die folgende Option **ENTER** drücken.

Symbole

Zum Ein– und Ausschalten der Markierung für die einzelnen Punkte in der Kurve. Funktioniert wie ein Kippschalter (Ein/Aus).

Symbol-Farbe

Zur Auswahl der Symbol-Farbe, analog zum Vorgehen bei der Linien-Farbe. Selbstverständlich kann die Option übersprungen werden, wenn keine Symbole verwendet werden sollen.

Gefüllte Linien

Schaltet hin und her zwischen einer gewöhnlichen Linie und einer bis zur horizontalen Achse gefüllten. Funktioniert ebenfalls in der Art eines Kippschalters (Ein/Aus).

Linienstärke

Zur Festlegung der Stärke. Geben Sie eine Zahl zwischen 0 (feinste) und 100 (stärkste) ein, gefolgt von **ENTER**. Wenn gefüllte Linien gewählt wurden, kann diese Option ausgelassen werden.

Zu guter letzt vergewissert sich Easel, ob Sie mit dem Ergebnis zufrieden sind. Bestätigen Sie dies mit **ENTER**, worauf Ihnen die neu definierte Grafik angezeigt wird, oder nehmen Sie gegebenenfalls weitere Änderungen vor.

Abbildung 7.1 wurde mit Format 2 (aufgestapelte Balken) erstellt und mit einer Zahlenserie in Linienform kombiniert.

NUMERISCHE EINGABE, LINIENDIAGRAMME

Wenn Sie für Ihre Werte eine Darstellung in Form eines Liniendiagramms wählen, ist die Erfassung der Daten genau so unkompliziert wie bei Balkendiagrammen – Sie geben die Werte ganz einfach über die Tastatur ein.

Der einzige wirkliche Unterschied zwischen Linien- und Balkendiagrammen zeigt sich bei der Eingabe von Werten in eine Zahlenserie, die in Linienform dargestellt ist. Easel erspart sich das komplette Neuzeichnen des Diagramms nach jeder Eingabe oder Änderung einer Zahl, indem es nicht die tatsächliche Linienfarbe benutzt.

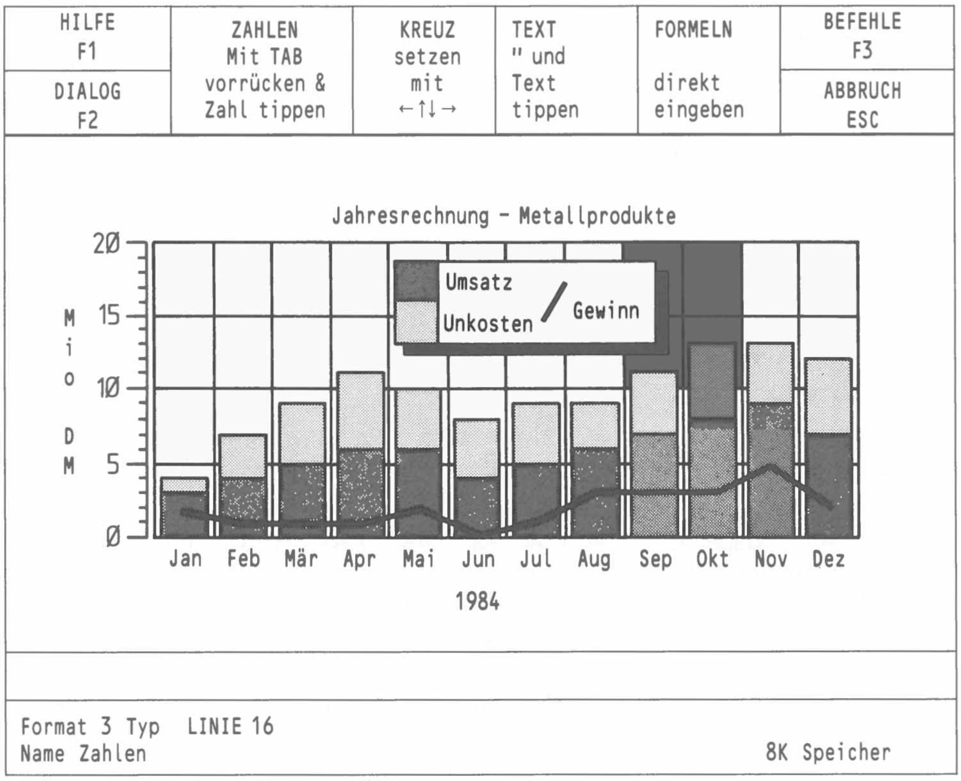


Abbildung 7.1 Gestapelte Balken und Linien

Solange Sie Zahlen eingeben, wird das Diagramm mit einer feinen weißen Linie (bzw. einer gefüllten Linie, falls Sie diesen Stil vorgesehen haben) gezeichnet. Die Farbe dieser Linie wechselt beim Durchqueren von Balken, Linien oder Text. Easel macht Sie im Statusbereich auf diesen Umstand aufmerksam ("Falsche Linienfarbe").

Wenn Sie mit der Zahleneingabe fertig sind, rufen Sie mit **F3** und **Z** den **Zeigen**-Befehl auf. Jetzt sehen Sie Ihr neues Diagramm in der gewünschten Farbe und Linienstärke.

KAPITEL 8
KREIS-DIAGRAMME

Obwohl ein Kreisdiagramm optisch nicht viel Ähnlichkeit hat mit einem Balken– oder einem Liniendiagramm, ändert sich am Arbeitsablauf nichts. Sie fordern ganz einfach das Grafikformat Nr. 7 an, und Easel verwandelt Ihre Zahlen in eine Kuchengrafik.

Es sei gleich gesagt, daß bei der Kreisdiagrammdarstellung jeweils nur eine Zahlenserie angezeigt werden kann und daß negative Werte nicht berücksichtigt werden. Easel meldet in solchen Fällen das Entfallen der betreffenden Daten.

Da, wie gesagt, nur immer eine Zahlenserie dargestellt werden kann, offeriert der **Zeigen**-Befehl die Option, die aktuellen Werte anzuzeigen, und nicht wie sonst "Alle Zahlenbilder". Ersetzen Sie den Namen gegebenenfalls durch einen anderen. Falls Sie eine Liste von Namen (getrennt durch Kommas) einsetzen, zeigt Easel das erste Zahlenbild in der Liste an und ignoriert den Rest.

NUMERISCHE EINGABE

Damit Sie sehen, wie die Eingabe von Zahlen bei einem Kreisdiagramm funktioniert, verwenden Sie den **Wechseln**-Befehl, wählen dort die Option "Form" und dann das Format Nr. 7 für Kreisdiagramme. Anschließend rufen Sie den Befehl **Neudaten** zur Eingabe einer ganz neuen Zahlenserie.

Geben Sie dem neuen Zahlenbild z.B. den Namen "Kosten". Easel zeichnet einen Kreis, welcher mit der ersten Beschriftung versehen ist – normalerweise "Jan".

Geben Sie eine Zahl und danach **ENTER** ein, worauf Easel den Kreis erneut zeichnet, diesmal unter Angabe der betreffenden Zahl direkt unter dem Namen. Damit haben Sie ein Kreisdiagramm mit einem einzigen Wert. Geben Sie noch ein paar weitere Zahlen ein, in der gleichen Weise wie bei den Balkendiagrammen.

Während der Dateneingabe in ein Kreisdiagramm wird das nächste Eingabefeld durch Hervorhebung seines Namens gekennzeichnet und, falls dies nicht möglich ist, durch ein spezielles Feld in der unteren linken Ecke des Anzeigebereichs. In Abbildung 8.1 ist das der Name "PORTUGAL".

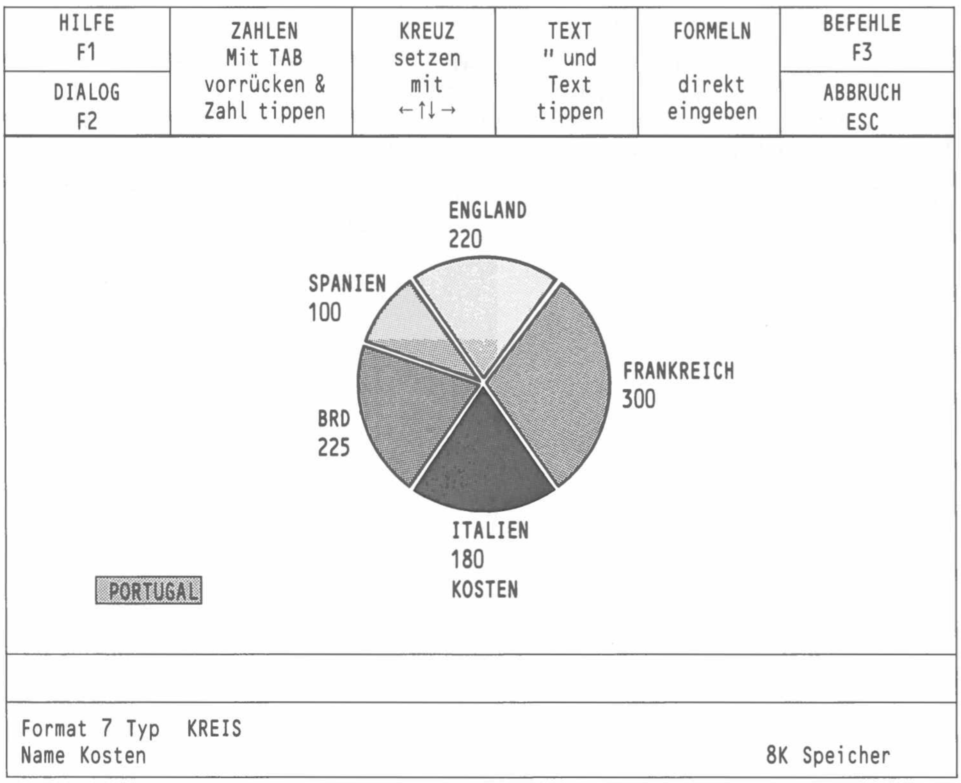


Abbildung 8.1 Ein Kreisdiagramm (auch Torten– oder Kuchendiagramm genannt)

Da der Kuchen bei jeder Eingabe oder Änderung einer Zahl als ganzes neu gezeichnet wird, ist es unter Umständen praktischer, die Werte in einem anderen Format einzutragen, und erst später auf Tortendarstellung überzuwechseln.

FELDNAMEN

Die Bewegung von einem Feld zum nächsten erfolgt mit **TAB** bzw. mit **SHIFT** und **TAB** (rückwärts), genau wie bei den Balkendiagrammen. Wie dort ist im Befehlsmenü keine Bewegung möglich. Kehren Sie also gegebenenfalls mit **ESC** zur Hauptanzeige zurück.

Mit **F5** schieben Sie direkt nach dem aktuellen ein neues "unbenanntes" Feld ein. Tippen Sie eine beliebige Zahl ein. Die Feldbezeichnung "unbenan" können Sie über die Option **B** im **Bearbeiten**-Menü editieren. Bearbeitet werden kann immer nur das aktuelle, d.h. das hervorgehobene Feld. **TAB** bzw. **SHIFT** und **TAB** bringen Sie von einer Bezeichnung zur nächsten.

Genau wie bei Balken- und Liniendiagrammen setzt das Löschen eines Feldes voraus, daß sowohl die Feldbezeichnung (**Bearbeiten**, **B** und **F4**) wie auch der dort befindliche Wert gelöscht wurden (mit **TAB** und **F4**). Easel löscht leere Felder beim nächsten **Zeigen**-Befehl.

TEXT

Das Hinzufügen, Löschen und Verschieben von Text und Titeln funktioniert wie bei den Balken- und Liniendiagrammen, gemäß den Anleitungen in den Kapiteln 2 und 4. Für Eingabe, Löschen und Verschieben von gewöhnlichem Text verwenden Sie wie dort das Fadenkreuz.

DER WECHSELN-BEFEHL

Die beiden Optionen "Text" und "Form" im **Wechseln**-Menü funktionieren in derselben Weise wie bei den Balkendiagrammen.

Die Optionen "Koordinatenpapier", "Linie" und "Achse" sind selbstverständlich im Zusammenhang mit Kreisdiagrammen unsinnig, und Easel erlaubt ihre Benutzung nicht.

Dafür gibt es die Sektor-Option, welche ausschließlich für Kreisdiagramme gilt. Sie dient zum Wechseln der Farbe eines Kreissektors, und wird über **Wechseln** und **S** aufgerufen. Zur Auswahl des Sektors drücken Sie solange **TAB**, bis die betreffende Bezeichnung durch Hervorhebung aktiviert ist.

Darauf erfolgt ein Bildschirmwechsel: Easel präsentiert im Anzeigebereich eine Farbpalette, aus der Sie mit Hilfe der Rechts- und Linkspfeiltasten die gewünschte Farbe auswählen und zur Bestätigung **ENTER** drücken. Darauf zeichnet Easel das Kreisdiagramm neu, mit dem betreffenden Sektor in der neuen Farbe.

KAPITEL 9

PAPIER-BILDER VON GRAFIKEN

DRUCKAUSGABE

Mit Hilfe eines Punktmatrixdruckers, wie mit dem QL Drucker oder einem Epson FX80 kompatiblen, können Sie Ihre Grafiken sofort auf Papier verewigen.

Der **Ausdruck**-Befehl gibt die Grafik, die auf dem Bildschirm angezeigt wird, an den Drucker aus. Zum Aufrufen wählen Sie aus dem Befehlsmenü **D** für **Ausdruck**. Easel liest dann das Druckertreiberprogramm "gprint_prt" von der Kassette im Microdrive 1.

Die Option **A** im Untermenü leitet die Ausgabe auf eine Microdrive-Datei, deren Namen Sie eingeben müssen. (Danach **ENTER** drücken.) Diese Datei kann dann beispielsweise mit Hilfe eines SuperBASIC-Programms verarbeitet und an einen Drucker geschickt werden, der von Easel nicht unterstützt wird. Hier soll erwähnt werden, daß eine solche Datei im allgemeinen sehr umfangreich ist, so daß maximal 3 auf eine Microdrive-Kassette passen.

Die Option **I**(nstallation) ist zu wählen, wenn ein neuer Druckertreiber installiert werden soll. Die Easel-Kassette enthält eine ganze Reihe verschiedener Druckertreiber. Sie erkennen die Dateien am Nachnamen **_prt**. Einige davon sind Farbdrucker, z.B. der Integral 132 und der Okimate 80. Geben Sie den entsprechenden Namen ein und drücken Sie **ENTER**.

Der neue Druckertreiber wird nicht permanent installiert, sondern nur für die momentane Arbeit. Beim Neustarten von Easel wird wiederum der Druckertreiber für den QL Drucker geladen, der in der Standard-Druckertreiberdatei "gprint_prt" enthalten ist. Auf Wunsch können Sie natürlich diese Standardvorgabe ändern, indem Sie die Druckertreiberdateien umbenennen: Kopieren Sie die ursprüngliche "gprint_prt"-Datei auf einen anderen Namen, z.B. "QL_prt" und löschen Sie dann "gprint_prt". Erstellen Sie dann eine Kopie des Druckertreibers, den Sie ständig verwenden wollen, und speichern Sie diese unter dem Namen "gprint_prt". Denken Sie daran, daß die Easel-Original-Kassette mit Schreibschutz versehen ist, so daß Sie für solche Operationen Ihre Reservekopie verwenden müssen.

Sie brauchen nicht notwendigerweise die QL-Standard-Baudrate von 9600 zu verwenden. Genausogut können Sie mit 4800 arbeiten, indem Sie zum Starten von Easel folgende Eingabe vornehmen:

```
BAUD 4800
LRUN mdv1_boot
```

statt die Easel-Kassette beim Drücken von **F1** oder **F2** im Microdrive 1 zu haben.

Wenn die Baudrate permanent geändert werden soll, empfiehlt es sich, dem Boot-Programm eine Zeile hinzuzufügen. Zu diesem Zweck laden Sie das Programm und numerieren es neu:

```
LOAD mdv1_boot
RENUM 10,10
```

Setzen Sie dann z.B. die Programmzeile

```
5 BAUD 4800
```

ein, löschen Sie die ursprüngliche Version und sichern Sie die neue auf der Easel-Kassette mit:

```
SAVE mdv1_boot
```

Auch diese Änderung können Sie nur auf Ihrer Easel-Reservekopie vornehmen.

PHOTOGRAPHIE

Die einfachste und schnellste Methode, eine beständige Kopie einer Grafik herzustellen, ist mit Hilfe eines Photoapparates. Für gute Ergebnisse muß allerdings mit etwas Sorgfalt vorgegangen werden.

Eine der häufigsten Ursachen für schlechte Photos ist eine zu kurze Belichtungszeit. Das Bild setzt sich aus 625 separaten Linien zusammen, von denen eine nach der anderen angezeigt wird. Die Anzeige aller Zeilen dauert 1/25 s. Wenn eine Belichtungszeit gewählt wird, die etwa gleich lang oder kürzer ist, resultiert ein ungleichmäßig belichtetes Bild. Am besten ist es, eine Belichtungszeit von etwa einer Viertelsekunde zu verwenden, was voraussetzt, daß Sie die Kamera auf ein Stativ montieren. Ein normaler Negativ-Farbfilm oder ein Diapositiv-Film mit einer Empfindlichkeit von etwa 100 ASA erfordert eine Blende von ungefähr f:5,6. Verwenden Sie nach Möglichkeit ein Teleobjektiv (etwa 100 mm), um die durch die Bildschirmwölbung verursachte Verzerrung zu verringern.

Machen Sie das Photo am besten in einem verdunkelten Raum. Auf diese Weise verhindern Sie Reflexionen auf der Bildschirmoberfläche. Es ist überraschend, wie störend solche Reflexionen in Photos wirken, selbst wenn man sie im Sucher gar nicht bemerkt.

Durch Drücken von **F2** schalten Sie den Steuerbereich aus und erhalten eine größere Grafikfläche. Mit **SHIFT** + **F3** können Sie die Anzeige im Statusbereich löschen. Vergewissern Sie sich, daß alle Beschriftungen, Titel und die Legende genau so sind, wie Sie es haben wollen, bevor Sie auslösen.

KAPITEL 10

QL EASEL ÜBERSICHT

DIE FUNKTIONSTASTEN

Zusätzlich zu der allgemeingültigen Belegung der Funktionstasten **F1**, **F2** und **F3** verwendet Easel die Funktionstasten **F4** und **F5** wie folgt:

[F4] Löschen
- Text
- Bezeichnungen
- Zahlen
- Legende
- benutzerdefinierte Objekte

Hinweis: unter benutzerdefinierten Objekten versteht man Balken, Linien, Koordinatenpapier und Achsen.

[F5] Einfügen eines Feldes

DIE BEFEHLE

Die Befehle verschaffen Ihnen den Zugang zu den leistungsfähigen Grafikfunktionen. Es folgt eine alphabetische Zusammenstellung aller Easel-Befehle.

AUSDRUCK

Zum Ausdrucken der gegenwärtig am Bildschirm angezeigten Grafik.

Das Untermenü enthält drei Optionen:

D dient zum Ausdrucken der Grafik unter Verwendung des aktuellen Druckertreiberprogramms.

A leitet die Ausgabe an eine Datei auf Microdrive-Kassette (sog. "Screen-Dump"), deren Namen Sie eingeben.

I dient zur Installation eines neuen Druckertreibers. Easel wartet auf die Eingabe des neuen Dateinamens (Nachname **_prt**). Die Easel-Kassette enthält eine Reihe verschiedener Grafikdruckertreiber. Eine ausführlichere Beschreibung finden Sie in Kapitel 9.

BEARBEITEN

Dieser Befehl dient zum Ändern und Verschieben der Bezeichnungen und der Legende.

Vorgesehen sind die vier Optionen:

Text	Das Fadenkreuz hängt sich automatisch an das nächste Textstück, und in der Eingabezeile wird der Text zur Überarbeitung mit dem Zeileneditor freigegeben. Nach Drücken von **ENTER** können Sie den Text auf Wunsch an eine andere Stelle bewegen und die Position mit **ENTER** bestätigen.
Bezeichnungen	Das Fadenkreuz hängt sich an die nächste Feldbezeichnung, die dann in der Eingabezeile editiert werden kann. Anschließend **ENTER** drücken und gegebenenfalls Verschiebung vornehmen.
Legende	Das Legendenkästchen kann sofort mit Hilfe der Pfeiltasten verschoben werden. Die gewünschte Position mit **ENTER** fixieren, worauf die Legende neu eingezeichnet wird.
Achse	**V** oder **H** für vertikale oder horizontale Achse eingeben, worauf sich das Fadenkreuz auf den betreffenden Achsennamen positioniert, den Sie dann überarbeiten und verschieben können.

Nach der Bearbeitung erscheint aller Text in der Farbe und der Richtung, wie sie bei der zuletzt verwendeten **Text**-Option des **Wechseln**-Befehls festgelegt wurden. Die einzige Ausnahme bilden die Achsen-Namen, deren Richtung nicht verändert werden kann.

DATEIEN

Dieser Befehl ermöglicht es, bereits auf Microdrive-Kassette gespeicherte Easel-Dateien zu ändern und Datendateien an andere Psion QL-Programme zu übertragen.

Das Untermenü enthält die folgenden Optionen:

Reserve	Zum Erstellen einer Zweitkopie einer Easel-Datei. Easel bittet um den Namen der Originaldatei und der Kopie. **Es empfiehlt sich unbedingt, von allen Dateien Reservekopien anzufertigen, um sich gegen Datenverlust durch versehentliches Überschreiben oder Beschädigung der Kassette sowie gegen Fehler zu schützen, die ein Anwendungsprogramm löschen oder unbrauchbar machen könnten.** Wird auf die ausdrückliche Angabe eines Datei-Nachnamens (einer Erweiterung) verzichtet, nimmt Easel standardmäßig den Nachnamen **_grf** an. Die neue Datei übernimmt denselben Nachnamen wie die ursprüngliche, es sei denn, Sie spezifizieren einen anderen.
Löschen	Zum Löschen einer benannten Datei von Microdrive-Kassette. **Vorsicht** – Dieser Befehl kann nicht rückgängig gemacht werden und ist mit äußerster Sorgfalt zu benutzen. Nichts überstürzen!
Export	Exportiert eine benannte Datei, welche alle Zahlenserien enthält, die sich im Arbeitsspeicher befinden. Die Sicherung erfolgt in einer Form, die den Import durch QL Abacus oder QL Archive erlaubt. Import und Export werden im Anhang ausführlich beschrieben. Ohne ausdrückliche Angabe eines Nachnamens weist Easel Exportdateien den Zusatz **_exp** zu.
Import	Importiert eine benannte Datei und ermöglicht Easel das Einlesen und die grafische Ausgabe von Dateien, die aus QL Abacus und QL Archive exportiert wurden. Ohne ausdrückliche Angabe eines Nachnamens erwartet Easel den Nachnamen **_exp**.
Formatieren	Formatiert die Kassette in Microdrive 2 bzw. einem anderen, benannten Microdrive. Easels Vorschlag mdv2 akzeptieren bzw. einen anderen Bezeichner eingeben, z.B. mdv3. Easel bittet um Bestätigung der ausgewählten Option. **Vorsicht** – Der Formatiervorgang vernichtet alle auf der Kassette befindlichen Daten.

ELIMINIEREN

Löscht eine oder mehrere Zahlenserien aus der Grafik und vernichtet die Daten. Bei Wahl dieses Befehls fordert Easel eine Liste der Zahlenbilder an, die gelöscht werden sollen. Die einzelnen Namen sind durch Kommas zu trennen und mit **ENTER** abzuschließen. Drücken von **ENTER** ohne andere Eingabe führt zur Zerstörung der aktuellen Zahlen. Die allgemeine Angabe "Eliminieren alle Zahlenbilder" wird ebenfalls akzeptiert.

PRESENT

Dient zur Bestimmung des Bildschirmformats: 40, 64 oder 80 Spalten pro Zeile. 4, 6 oder 8 drücken.

HERVORHEBEN

Mit diesem Befehl können Sie zur Hervorhebung einer bestimmten Zahl innerhalb eines Zahlenbildes – oder beispielsweise auch für alle negativen Werte in einem Balkendiagramm – einen speziellen Stil wählen.

Zuerst bittet Easel, **W** zur Hervorhebung eines bestimmten Wertes bzw. **N** für negative Werte zu drücken. Letztere Option ist natürlich bei Kreisdiagrammen nicht gestattet.

Nach Eingabe von **W** fordert Easel die Wahl des Wertes. Drücken Sie **TAB** (bzw. **SHIFT** und **TAB**) zum Markieren des betreffenden Feldes, gefolgt von **ENTER**. Bei Balkendiagrammen präsentiert Easel die Auswahl der vordefinierten Balken. Wählen Sie einen aus oder entwerfen Sie einen neuen. Bei Kreisdiagrammen wird das ausgewählte Segment aus dem Kuchen "herausgeschnitten" (s. Abbildung 10.1).

Wenn Sie die Option **N** für Hervorhebung negativer Werte wählen, bittet Easel sofort um Auswahl eines Balkens.

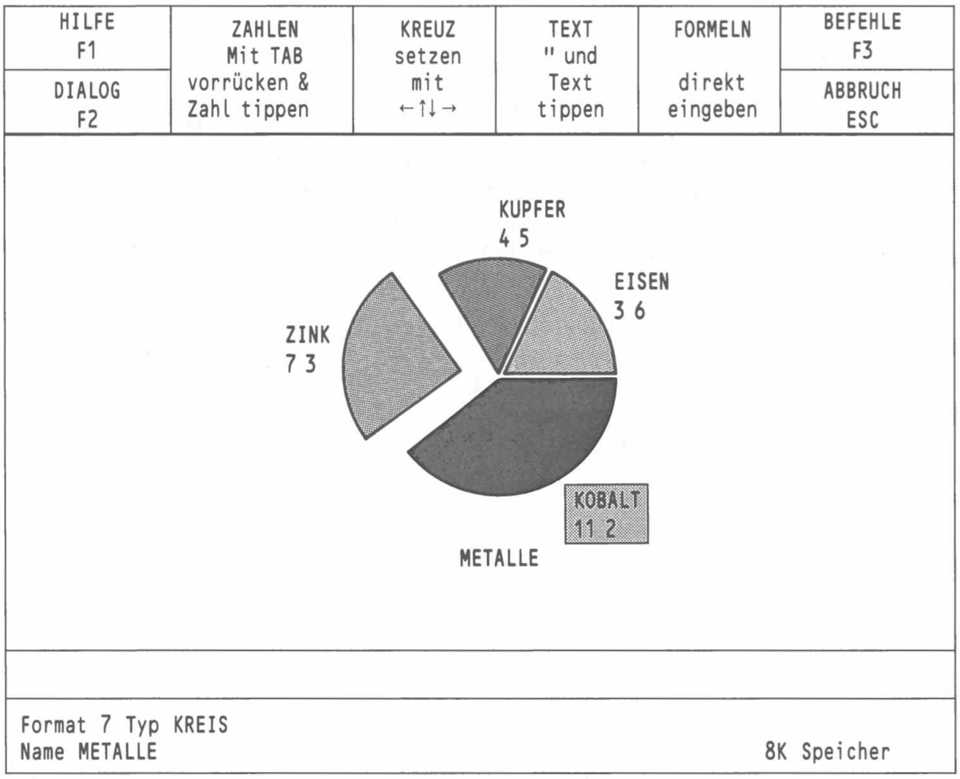


Abbildung 10.1 Hervorgehobener Sektor im Kreisdiagramm

ISTDATEN

Mit diesem Befehl machen Sie ein bestimmtes Zahlenbild zum aktuellen. Easel bittet um Eingabe des gewünschten Namens (keine Anführungszeichen notwendig). Sofort nach Drücken von **ENTER** befinden Sie sich im Dateneingabemodus und können beliebige Änderungen vornehmen.

LADEN

Lädt ein bereits auf Microdrive-Kassette gesichertes Zahlenbild in den Arbeitsspeicher. Sie werden um den Namen gebeten. Gleichzeitig mit den Daten werden alle Format-Optionen eingelesen, so daß das Zahlenbild exakt so geladen wird, wie es vor dem Speichern am Bildschirm angezeigt wurde.

Ohne ausdrückliche Erwähnung eines Namenszusatzes nimmt Easel den Nachnamen **_grf** an.

NEUDATEN

Dient zum Erstellen einer neuen Zahlenserie, die zum aktuellen Zahlenbild wird. Easel fordert die Eingabe eines Namens (keine Anführungszeichen notwendig). Mit **ENTER** gelangen Sie in den Dateneingabemodus und können beliebige Werte eintragen.

SICHERN

Sichert alle auf dem Bildschirm angezeigten Zahlenbilder auf Microdrive-Kassette. Easel fordert die Eingabe eines Dateinamens und nimmt als Namenszusatz **_grf** an, wenn Sie nicht ausdrücklich etwas anderes spezifizieren.

TILGEN

Dieser Befehl löscht alle Zahlenbilder, alle Beschriftungen und alle benutzerdefinierten Objekte (Balken, Linien usw.) und kehrt zu den ursprünglichen Feldbezeichnungen (Monate) zurück.

UMBENENNEN

Dieser Befehl weist einem bereits bestehenden Zahlenbild einen neuen Namen zu. Easel fordert die Eingabe des ursprünglichen Namens, wobei es das aktuelle Zahlenbild vorschlägt, und danach den neuen Namen. Nach jedem Namen ist jeweils **ENTER** zu drücken.

VERLASSEN

Dieser Befehl dient zum Aussteigen aus Easel und zur Rückkehr in SuperBASIC. Ausgeführt wird er nach Bestätigung mit **ENTER**; falls Sie Ihre Entscheidung noch rückgängig machen wollen, drücken Sie **ESC** und landen wieder im Befehlsmenü.

WECHSELN

Ein äußerst vielseitiger Befehl zum Ändern aller Aspekte Ihrer Grafik.

Das Untermenü enthält die folgenden Optionen:

Achse Zum Ändern der Achsenteilstriche, der Farbe und der numerischen Beschriftung. Die Achsenstriche können eingezeichnet werden oder unsichtbar bleiben. Nicht gültig ist diese Option in Format 7 (Kreisdiagramm).

Bei den Achsenbegrenzungen wählen Sie zwischen automatischer und manueller Begrenzung, **A** ist für automatisch und variabel, **N** für automatisch unter Einschluß des Nullpunkts.

Alternativ kann **M** gewählt werden, wobei Sie beide Grenzwerte von Hand eingeben. Bei Bedarf ändert Easel diese Werte, um eine Darstellung aller Zahlen anhand von einfachen numerischen Skalenwerten zu ermöglichen.

Balken Zur Auswahl bzw. zur eigenen Definition der Balken zur Darstellung des aktuellen Zahlenbildes. Easel verfügt über 16 vordefinierte Balken, die über ihre Nummer oder anhand von Beispieldiagrammen ausgewählt werden können. Letztere Methode bietet zusätzlich die Möglichkeit, einen eigenen Balkenstil zu entwerfen. Nicht gültig ist diese Option in Format 7 (Kreisdiagramm).

Format Zur Neudefinition des gesamten Diagramms. Easel kennt 8 verschiedene Formate, die Sie durch Eingabe einer Nummer oder anhand von Beispielgrafiken auswählen.

Koordinatenpapier Zur Wahl stehen 7 verschiedene Arten von Koordinatenpapier und die Möglichkeit, ein eigenes zu entwerfen. Variable Aspekte sind Hintergrund– und Vordergrundfarbe. Die Option ist nicht zulässig, wenn in Format 7 (Kreisdiagramm) gearbeitet wird.

Linie Es gibt 16 verschiedene, vordefinierte Stile und die Möglichkeit, einen eigenen zu entwerfen. Variable Aspekte sind Linienfarbe und –stärke, Symbol für die Teilpunkte in der Kurve oder "Auffüllen" des Bereichs zwischen dem Linienzug und dem Nullpunkt. Nicht zulässig ist diese Option beim Arbeiten in Format 7 (Kreisdiagramm).

Sektor Dient zum andersfarbigen Ausmalen eines bestimmten Kreissektors. Option gilt nur für Format 7 (Kreisdiagramm).

Text Zur Auswahl der Textfarbe und des Texthintergrunds, welcher auch durchsichtig sein kann, so daß die darunterliegende Grafik sichtbar bleibt. Text kann auf Wunsch horizontal oder vertikal angeordnet werden.

Die vorgenommenen Änderungen betreffen nur neu eingegebenen Text, während bereits bestehender unverändert bleibt. Der Text in der Legende nimmt immer die aktuelle Textfarbe an.

ZEIGEN

Dient zur Darstellung einer Grafik, die entweder alle vorhandenen oder eine Auswahl der Zahlenbilder vereinigt. Standardmäßig sieht Easel die Anzeige aller Zahlenbilder vor, was Sie mit **ENTER** akzeptieren oder durch selektive Eingabe einer Liste ändern können. Die Namen in einer Liste sind durch Kommas zu trennen und mit **ENTER** abzuschließen.

Bei Kreisdiagrammen schlägt Easel jeweils nur den Namen des aktuellen Zahlenbildes vor. Wenn Sie eine Liste eingeben, gelangt lediglich der erste Name zur Anzeige. Der Rest wird ignoriert.

Anschließend wird als Anzeigeformat das zuletzt verwendete vorgeschlagen, was Sie mit **ENTER** gutheißen oder durch eine andere Nummer und Drücken von **ENTER** ersetzen können.

FUNKTIONEN

Stellen Sie sich eine Funktion am besten als eine Art Rezept vor, das mit einem oder mehreren Werten, sog. *Argumenten*, eine bestimmte Operation ausführt und ein Ergebnis liefert, den *Funktionswert*.

In Easel gibt es Funktionen, die ein, zwei oder drei Argumente übernehmen, und auch solche, die ohne ein Argument auskommen. Die Argumente werden direkt nach der Funktion in Klammern angegeben. Zwischen dem Funktionsnamen und der Eröffnungsklammer darf kein Leerzeichen gelassen werden; hingegen sind Leerzeichen zwischen den Werten in der Klammer gestattet. Mehrere Argumente sind durch Kommas voneinander zu trennen. Allen Funktionen muß ein Klammerpaar folgen, auch wenn sie keine Argumente enthalten. Dies hilft, eine Funktion als solche zu identifizieren und Verwechslungen zwischen Variablen und Funktionen zu vermeiden, falls sie den gleichen Namen haben.

Es folgt eine alphabetische Zusammenstellung aller in Easel verfügbaren Funktionen.

ABS(n) — Liefert den absoluten Wert des Arguments d.h. den Wert ohne Berücksichtigung eines etwaigen Minuszeichens. **abs(5)** und **abs(-5)** liefern beide den Wert 5.

BOG(n) — Wandelt einen in Grad ausgedrückten Winkel in Bogenmaß um.

EXP(n) — Liefert den Wert von e (ca. 2,718) hoch n, wobei das Resultat verfälscht ist, wenn n außerhalb des Bereichs −87 bis +88 liegt, weil dies den numerischen Bereich von Easel übersteigt.

GANZZAHL(n) — Liefert den ganzzahligen Teil des Arguments und schneidet etwaige Dezimalstellen ab. Die Rundung erfolgt stets in Richtung Null, z.B.:

`ganzzahl(3.7)` liefert 3
`ganzzahl(-4.8)` liefert −4

GRAD(n) — Wandelt einen in Bogenmaß ausgedrückten Winkel in Grad um.

KOS(n) — Liefert den Kosinus des gegebenen Winkels (in Bogenmaß).

LN(n) — Gibt den natürlichen Logarithmus (zur Basis e) der Zahl n im Argument aus. Ein Wert kleiner oder gleich Null führt zu einer Fehlermeldung, da für diesen Wertebereich keine Logarithmen definiert sind.

PI() — Liefert den Wert der mathematischen Konstanten π.

QWURZEL(n) — Gibt die Quadratwurzel des Arguments n aus, welches keine negative Zahl sein darf.

SIN(n) — Gibt den Sinus des Winkels n (in Bogenmaß) als Resultat aus.

TAN(n) — Liefert den Tangens des angegebenen Winkels (in Bogenmaß).

VORZ(n) — Liefert +1 bei einem positiven, −1 bei einem negativen und 0 bei einem Nullargument.

Die folgenden beiden Elemente sind keine Funktionen, obwohl sie einen Wert ausgeben. Sie dürfen nur im Rahmen einer Formel eingesetzt werden, die ein neues Zahlenbild erstellt.

FELD — Innerhalb einer Formel liefert **Feld** die Nummer jedes Feldes. Es weist im ersten Feld den Wert 1, im zweiten 2 ... usw. auf. Ein einfaches Beispiel zum Anlegen eines Zahlenbildes unter dem Namen "Beispiel" wäre:

```
Beispiel = Feld
```

FELDMAX — Innerhalb einer Formel liefert **Feldmax** für jedes Feld die Anzahl der aktuell belegten Felder. Auf diese Weise ist es möglich, die Anzahl Felder in einem Zahlenbild durch Errichten einer neuen Zahlenserie zu ermitteln:

```
Umfang = Feldmax
```

KAPITEL 1
WAS IST EASEL? 1

KAPITEL 2

FANGEN SIE AN 2
QL EASEL LADEN 2
DIE BILDSCHIRMMASKE 2
Der Statusbereich 2
Der Anzeigenbereich 3
Der Steuerbereich 3
DAS FADENKREUZ 3
ZAHLEN 4
TEXT 4
FORMELN 4
DIE BEFEHLE 4
WERTE LÖSCHEN 5
WERTE EINFÜGEN 5

KAPITEL 3

GESTALTUNG VON BALKEN 6
EINE BALKENFORM WÄHLEN 6
Wahl mit Nummer 6
Wahl anhand von Beispielen 6
EIGENENTWURF 7
Balken-Farbe 7
Randfarbe 7
Randstärke 7

KAPITEL 4

VERWENDUNG VON TEXT 8
GEWÖHNLICHER TEXT 8
ACHSEN-NAMEN 8
FELDNAMEN 8
TEXTFARBE UND RICHTUNG 8
Textfarbe 9
Papierfarbe-Texthintergrund 9
Textrichtung 9

KAPITEL 5

MEHRERE ZAHLENBILDER 10
DIE AKTUELLEN ZAHLEN 10
DER UMBENENNEN-BEFEHL 10
DER NEUDATEN-BEFEHL 10
VERWENDUNG EINER FORMEL 10
DER ISTDATEN-BEFEHL 11
DATEN ZEIGEN 12
DIE LEGENDE 12

KAPITEL 6

GRAFIK-FORMATE 13
FORMATE NEU DEFINIEREN 13
Koordinatenpapier 13
Achse 13

KAPITEL 7

LINIEN-DIAGRAMME 14
VERSCHIEDENE LINIENARTEN 14
Linien-Farbe 14
Symbole 14
Symbol-Farbe 14
Gefüllte Linien 14
Linienstärke 14
NUMERISCHE EINGABE, ...
LINIEN-DIAGRAMME 15

KAPITEL 8

KREISDIAGRAMME 16
NUMERISCHE EINGABE 16
FELDNAMEN 17
TEXT 17
DER WECHSELN-BEFEHL 17

KAPITEL 9

PAPIER-BILDER VON GRAFIKEN 18
DRUCKAUSGABE 18
PHOTOGRAPHIE 18

KAPITEL 10

QL EASEL ÜBERSICHT 20
DIE FUNKTIONSTASTEN 20
DIE BEFEHLE 20
AUSDRUCK 20
BEARBEITEN 20
DATEIEN 20
ELIMINIEREN 21
PRESENT 21
HERVORHEBEN 21
ISTDATEN 22
LADEN 22
NEUDATEN 22
SICHERN 22
TILGEN 22
UMBENENNEN 22
VERLASSEN 22
WECHSELN 23
ZEIGEN 23
FUNKTIONEN 24

sinclair

QL

Information

by Dick de Grandis-Harrison *(Psion Limited)*

QL PROGRAMME –
IMPORT UND EXPORT

Die beiden Befehle **import** und **export** dienen zur Übertragung von Informationen zwischen den vier QL-Programmen. Die Form der Datenspeicherung ist bei QL Abacus, QL Archive und QL Easel jeweils ähnlich und gestattet eine Darstellung in tabellarischer Form. Aus diesem Grund ist der Austausch unkompliziert und geht reibungslos. In Abacus und Easel gehören **import** und **export** zu den Optionen im **dateien**-Untermenü; in Archive sind es zwei eigenständige Befehle.

Befassen wir uns zunächst mit dem Import und Export zwischen Abacus, Archive und Easel. Die von diesen drei Programmen erstellten Exportdateien sind in der Struktur identisch und können daher problemlos gegenseitig importiert werden.

Nehmen wir an, wir haben eine Abacus-Kalkulationstabelle mit den folgenden Informationen:

	A	B	C	D
1	Cash_flow$	Januar	Februar	März
2	Verkauf	1000	1050	1100
3	Kosten	500	530	560
4	Gewinn	500	520	540

Abacus-Tabelle zum Export

Beim Import werden diese Daten von Easel als drei Zahlenbilder namens *Kosten, Verkauf und Gewinn* interpretiert. Easel verwendet die Monatsnamen als Feldbezeichnungen für die Diagramme, so daß sich die Informationen wie folgt darstellen:

Cash_flow$	Januar	Februar	März
Verkauf	1000	1050	1100
Kosten	500	530	560
Gewinn	500	520	540

Import durch Easel

Keine Verwendung macht Easel von dem ersten Stück Text, *Cash__flow$*. Beim Exportieren eines Zahlenbildes aus Easel wird der Text, automatisch an dieser Stelle eingesetzt, um Kompatibilität zu wahren.

Wenn wir die gleiche Zahlenserie durch Archive importieren, ist das Ergebnis eine Datei mit drei Datensätzen, von denen jeder. Satz vier Felder enthält, nämlich *Cash__flow$* (Textfeld), *Verkauf*, *Kosten* und *Gewinn* (numerische Felder). Die Datei sieht so aus:

Felder	Datensatz 1	Datensatz 2	Datensatz 3
Cash_flow$	Januar	Februar	März
Verkauf	1000	1050	1100
Kosten	500	530	560
Gewinn	500	520	540

Import durch Archive

REGELN

Um den reibungslosen Datenaustausch zwischen den drei Programmen zu gestatten, müssen die folgenden Regeln beachtet werden:

1. Beim Exportieren einer Abacus-Tabelle muß jeweils das erste Feld jeder Zeile (bzw. jeder Spalte bei spaltenweisem Export) im betreffenden Tabellenausschnitt mit Text versehen sein.
2. Zeilen (bzw. Spalten), deren erstes Feld leer ist, werden nicht exportiert.
3. Das direkt auf das Textfeld folgende Feld einer zum Export vorgesehenen Zeile oder Spalte muß Daten enthalten. Der Datentyp im Rest der Zeile (bzw. Spalte) wird durch den dort vorkommenden Typ bestimmt. Eine Zeile (oder Spalte) darf immer nur einen Datentyp enthalten, also nur numerische oder nur Zeichenketten–Daten.
4. Aus Abacus und Archive können Dateien mit mehreren Textdaten-Serien exportiert werden; von Easel exportierte Dateien enthalten jedoch immer nur eine Textserie, die Feldbezeichnungen.
5. Beim Import einer Datei mit mehr als einer Textserie verwendet Easel die erste als Feldbezeichnungen und ignoriert die übrigen.

6 Zum Export einer Abacus-Tabelle an Archive muß das erste Feld einer Zeile (oder Spalte bei spaltenweisem Export) nach den Regeln für Archive-Namen gebildet sein. Allerdings dürfen Bezeichnungen für Textfelder nicht mit einem $-Zeichen enden, weil diese ggfs. von Abacus automatisch eingesetzt werden.

DATEIENSTRUKTUR

Die Struktur einer Exportdatei besteht aus einer Reihe von Datensätzen, von denen jeder durch <**CR**> (ASCII Code 13) und <**LF**> (ASCII Code 10) abgeschlossen wird. Allerdings akzeptieren die Importbefehle diese Codes sowohl einzeln als auch zusammen, wobei die Reihenfolge unwichtig ist. Das Ende der Datei wird durch **CTRL Z** markiert (ASCII Code 26).

Jeder Datensatz besteht aus einer Reihe von Werten, die durch Kommas voneinander getrennt sind. Die Werte können sowohl Text (begrenzt durch Anführungszeichen) als auch Zahlen sein.

Jeder Datensatz muß mit einem Textwert beginnen und, wenn sein Name mit einem Dollarzeichen endet, müssen notwendigerweise auch alle folgenden Werte vom Texttyp sein.

Unser vorheriges Beispiel einer Abacus-Tabelle würde die folgende Exportdatei erzeugen:

```
"Cash_flow$","Verkauf","Kosten","Gewinn"<LF>
"Januar",1000,500,500<LF>
"Februar",1050,530,520<LF>
"März",1100,560,540<LF><CTRL Z>
```

Eine Exportdatei

Es ist auch möglich, Exportdateien aus SuperBASIC zu erstellen. Das folgende Programm erzeugt eine Exportdatei namens *Beispiel_exp,* für die bereits bekannten Daten.

```
100 OPEN_NEW#4,mdv2_Beispiel_exp
120 PRINT #4,'"Cash_flow$","Verkauf","Kosten","Gewinn"'
130 PRINT #4,'"Januar",1000,500,500'
140 PRINT #4,'"Februar",1050,530,520'
150 PRINT #4,'"März",1100,560,540'
160 PRINT #4, CHR$(26)
170 CLOSE #4
```

SuperBASIC setzt automatisch ein Steuerzeichen für Zeilenvorschub (ASCII Code 10) ans Ende jedes Datensatzes.

EXPORT NACH QUILL

QL Quill arbeitet mit formatiertem Text und erwartet auch bei Dateien, die aus den anderen QL-Programmen übernommen werden, nicht das gewöhliche Exportformat, sondern formatierten Text. Quill akzeptiert jeden Text, der Formularvorschübe (ASCII Code 12), Zeilenvorschübe (ASCII Code 10) und darstellbare Zeichen enthält, wobei Zeilenvorschübe als "Absatzende" und Formularvorschübe als "Seitenende" interpretiert werden. Etwaige andere Zeichen werden nicht beachtet.

Quill fragt beim **import**-Befehl ob Sie per Zeile oder per Absatz importieren wollen. Wenn Sie Import per Zeile wählen, so beginnt Quill mit jeder Zeile einen neuen Absatz. Wählen Sie dagegen Import per Absatz, so beginnt Quill nur dann einen neuen Absatz, wenn mehrere Zeilenvorschübe aufeinanderfolgen. Ein einzelner Zeilenvorschub erzeugt dann im importierten Dokument nur ein Leerzeichen.

Abacus und Archive können speziell für den Import durch Quill vorgesehene Dateien erstellen. Archive tut dies in Form des sog. formatierten Reports anhand von **Drucken**. Exportiert wird der Report dann als Druckausgabe über eine Microdrive-Datei mit der **Export**– Option des **Spulen**-Befehls. (Vgl. Kapitel 12 des Archive– Handbuchs.)

QL PROGRAMM – DRUCKAUSGABE

Die Original QL-Programmkassette ist schreibgeschützt und für die Druckerinstallationsprozedur nicht verwendbar. Fertigen Sie eine Arbeitskopie an, und legen Sie diese in den Microdrive.

Die vier Psion QL-Programme funktionieren mit allen Druckermodellen, die mit einem RS-232-C Interface ausgerüstet sind.

Sie können sowohl Endlos– wie auch Einzelblätter in den Drucker einspannen. Bei Verwendung von Einzelblättern hält der Drucker jeweils am Seitenende an, während am Bildschirm eine entsprechende Meldung angezeigt wird. Drücken Sie **ENTER** zum Weitermachen bzw. **ESC** zum Abbrechen des Druckvorgangs.

DRUCKERTREIBER

Der Drucker wird durch ein spezielles Programm gesteuert, den **Druckertreiber**, welcher anhand einiger Modifikationen dem Drucker angepaßt werden kann, der Ihrem System angeschlossen ist.

Steuerzeichen, mit Ausnahme der Codes für Zeilenvorschub und Wagenrücklauf, müssen durch einen ASCII-Code 0 (**NULL**) angeführt werden, damit der Drucker sie als Ausgabe erkennt. Beispielsweise ist das Steuerzeichen für verstärkte Schrift (Fettschrift) beim QL Printer der ASCII Code 27 (**ESC**) und **E**. Da **ESC** kein Druckzeichen ist, muß ihm eine ASCII-**NULL** vorangestellt werden. Aus Archive können Sie die Codes mit dieser Anweisung schicken:

```
drucken zchn(0) + zchn(27) + "E"
```

und in Abacus geben Sie:

```
zchn(0) + zchn(27) + "E"
```

in das Feld ein, bei dem mit Fettdruck begonnen werden soll.

Die Anpassung von QL Quill, QL Abacus und QL Archive an andere Drucker nennt man "Installieren der Software". Zu diesem Zweck gibt es ein spezielles SuperBASIC Installationsprogramm. Dieses Programm, **install__bas**, sowie Installationsdaten für eine Reihe von Druckern (**install__dat**) und die Installationsdaten für den angeschlossenen Drucker (**printer__dat**) befinden sich auf der Kassette "Utilities". Das Programm kann auch zum Anschluß eines Druckers für QL Archive dienen.

Die Abacus, Archive und Quill Programme selbst verwenden ausschließlich die Angaben in **printer__dat**. Die Datei "printer__dat" befindet sich auf jeder Programm-Kassette.

INSTALLATION EINES SERIELLEN DRUCKERS

Um einen Drucker mit serieller Schnittstelle an Quill anzupassen, legen Sie die Utilities Kassette in Microdrive 1, die Quill Kassette in Microdrive 2 und geben Sie ein

```
lrun mdv1_install_bas
```

worauf das Installationsprogramm anfängt zu laufen. Es setzt das Vorhandensein von **"install__dat"** voraus.

Als erstes wählen Sie den Microdrive für die Installation des Druckers. Drücken Sie in diesem Fall 2, gefolgt von **ENTER**, zur Installation in Microdrive 2. Mit **ENTER** wählen Sie einen seriellen Drucker (zum Betrieb über den seriellen Anschluß **ser1** oder **ser2**).

Das Programm liest dann die Installationsdaten und zeigt eine Liste der Drucker an, für die ein maßgeschneiderter Druckertreiber verfügbar ist.

Zur Auswahl des gewünschten Druckers verwenden Sie die Auf– und Abwärtspfeiltasten. Sobald der richtige Drucker hervorgehoben ist, drücken Sie **F5**. Die endgültige Installation wird erst vorgenommen, wenn Sie die Wahl mit **ENTER** bestätigen. Zum Annullieren und zur Rückkehr in die Liste der verfügbaren Drucker schlagen Sie eine beliebige Taste an.

Nach erfolgter Installation befinden Sie sich wieder in SuperBASIC. Wenn Sie Quill das nächste Mal laden, ist es auf den installierten Drucker und auf dessen Sonderfunktionen (Fettschrift, Unterstreichung, Hoch– und Tiefstellung) eingerichtet.

Wenn Sie für QL-Abacus oder QL-Archive einen Druckertreiber installieren wollen, gehen Sie genauso vor. Sie müssen lediglich die Kassette des entsprechenden Programms in Microdrive 2 einlegen.

Zum Entfernen eines Druckers aus der Liste dient **F3** und zum Speichern aller Druckertreiber **F4**. Da diese Funktionen unwiderrufliche Änderungen an den Druckertreiberinformationen vornehmen, werden sie erst nach Drücken von **ENTER** ausgeführt.

ANDERE SERIELLE DRUCKER

Bei Verwendung eines Druckers, der nicht in der Liste enthalten ist, kann auf zwei Arten vorgegangen werden:

Gar nichts tun

Verlassen Sie das Installatinsprogramm mit **ESC**. Alle vier QL-Programme verfügen bereits über einen eingebauten Druckertreiber, der gewöhnliche Textausgabe auf praktisch jedem Drucker ermöglicht.

Drucker hinzufügen

Sie können Ihren Drucker in die Liste aufnehmen. Dazu gibt es drei Methoden:

1. Wählen Sie mit dem Abwärtspfeil die Zeile "ANDERE". Drücken Sie **F1** oder **F2** zum Einrichten eines neuen speziellen Druckertreibers.
2. Wählen Sie einen Drucker aus der Liste und drücken Sie **F1** um eine Kopie des gewähten Programms zu erzeugen. Diese Option bietet sich an, wenn Ihr Drucker große Ähnlichkeit mit einem aus der Liste aufweist.
3. Wählen Sie einen vorliegenden Druckertreiber und drücken Sie **F2**. Auf diese Weise wird keine Kopie erstellt, sondern Sie nehmen die Änderungen direkt im ursprünglichen Programm vor. Verwenden Sie diese Methode wirklich nur, wenn Sie ganz genau wissen, was für Änderungen Sie vornehmen wollen.

In allen Fällen wird Ihnen eine Liste mit Druckerparametern zur Veränderung vorgelegt. Wählen Sie die einzelnen Parameter mit den Auf– und Abwärtspfeiltasten an und verwenden Sie den Rechts– und den Linkspfeil zum Ausführen der Modifikationen.

Die Liste enthält zwei Arten von Parametern:

– solche, die viele verschiedene Werte zulassen, wie etwa DRUCKERNAME und ZEILENENDE-CODE,

– und solche mit einer geringen Anzahl von möglichen Werten, wie PARITÄT.

	Standard	Andere Werte
TREIBERNAME	: STANDARD	....
ANSCHLUSS	: ser1	ser2
BAUD-RATE	: 9600	75, 300, 600, 1200, 2400, 4800
PARITÄT	: NICHTS	SPACE,MARK,UNGERADE,GERADE
ZEILEN/SEITE	: 66	0 bis 255
ZEICHEN/ZEILE	: 80	0 bis 255
ENDLOSFORMULARE	: JA	NEIN
ZEILENENDE-CODE	: CR,LF	....
PRÄAMBLE-CODE	: NICHTS	....
POSTAMBLE-CODE	: NICHTS	....
FETTDRUCK EIN	: NICHTS	....
FETTDRUCK AUS	: NICHTS	....
UNTERSTR. EIN	: NICHTS	....
UNTERSTR. AUS	: NICHTS	....
TIEFSTELLEN EIN	: NICHTS	....
TIEFSTELLEN AUS	: NICHTS	....
HOCHSTELLEN EIN	: NICHTS	....
HOCHSTELLEN AUS	: NICHTS	....
ÜBERSETZEN 1	: NICHTS	....
ÜBERSETZEN 2	: NICHTS	....
ÜBERSETZEN 3	: NICHTS	....
ÜBERSETZEN 4	: NICHTS	....
ÜBERSETZEN 5	: NICHTS	....
ÜBERSETZEN 6	: NICHTS	....
ÜBERSETZEN 7	: NICHTS	....
ÜBERSETZEN 8	: NICHTS	....
ÜBERSETZEN 9	: NICHTS	....
ÜBERSETZEN 10	: NICHTS	....

Diese beiden Parametertypen werden verschieden geändert. Die obenstehende Tabelle zeigt die Werte des STANDARD-Druckers. In der Kolonne rechts werden andere mögliche Werte (bei den Parametern mit geringer Auswahl) aufgeführt.

Bei den Parametern mit wenigen Auswahlmöglichkeiten verändert sich der Wert mit jedem Druck auf den Rechts- oder den Linkspfeil.

Bei den übrigen Parametern bewirkt das Drücken des Rechts- oder Linkspfeils, daß der bestehende Wert gelöscht wird. Geben Sie dann den Wert ein, der auf Ihren Drucker zutrifft, und drücken Sie **ENTER**. Alle Parameter dieses Typs, mit Ausnahme von DRUCKERNAME, akzeptieren Listen mit bis zu zehn durch Kommas voneinander getrennte Codes. Die Eingabe der Codes kann auf verschiedene Art erfolgen:

1. Als Zahl zwischen 0 und 255
2. Als Hex-Zahl, mit führendem Dollarzeichen, zwischen $0 und $FF
3. Als Einzelzeichen, mit führendem Anführungszeichen (" oder ')
4. Als standardmäßiges ASCII-Steuercode-Mnemonik, in Groß- oder Kleinschreibung:

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
BS	HT	LF	VT	FF	CR	SO	SI
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
CAN	EM	SUB	ESC	FS	GS	RS	US

5. Die Zeichenfolge FEHLW (bzw. fehlw) bewirkt eine Aktion, die beim Drucker einen Rückschritt auslöst. Sie sollte ausschließlich für die Verstärkungs- und Unterstreichungsfunktion verwendet werden und muß immer paarweise auftreten. Wenn z.B. UNTERSTREICHUNG EIN auf FEHLW gesetzt wird, muß dasselbe für UNTERSTREICHUNG AUS gelten. Die Verwendung von FEHLW setzt voraus, daß der Drucker richtig auf den Rückschritt-Code reagiert.

Drücken von **ENTER** ohne Eingabe wählt "NICHTS", d.h. verzichtet auf die Funktion. Die beschriebenen Eingabemethoden können beliebig kombiniert werden.

Der DRUCKERNAME besteht aus dem Namen des Herstellers oder des Modells. Er dient zur Identifikation der einzelnen Druckertreiber und darf nicht länger als 16 Zeichen sein. Zum Ändern den Rechts- oder Linkspfeil drücken und den gewünschten Namen, gefolgt von **ENTER** eingeben.

Der ANSCHLUSS ist ser1 oder ser2 zur Auswahl eines der beiden Standardanschlüsse.

Die BAUD-RATE bestimmt die Übertragungsgeschwindigkeit über die serielle Schnittstelle (Interface), ausgedrückt als die Anzahl von Bits pro Sekunde. 110 Baud entspricht etwa 10 Zeichen pro Sekunde, 300 Baud etwa 30 Zeichen pro Sekunde, usw. Die Baud-Rate des Druckertreibers muß mit dem seriellen Interface Ihres Druckers übereinstimmen.

Die PARITÄT ist abhängig von der Art, wie der Drucker das erste signifikante Bit (Binärziffer) in den vom Computer übertragenen Daten behandelt. Alle ASCII-Codes liegen im Bereich zwischen 0 und 127 und lassen sich als 7-stellige Binärziffer darstellen. Viele serielle Drucker erwarten die Zeichen im 7-Bit-Format. Daneben gibt es andere, die 8-Bit-Worte und folglich Codes zwischen 0 und 255 akzeptieren. Die zusätzlichen Codes im Bereich 128 bis 255 können als Graphikzeichen oder als akzentuierte Buchstaben ausgedruckt werden. Unter Umständen interpretiert Ihr Drucker das achte Bit eines 8-Bit-Codes als das *Paritätsbit* (auch *Prüfbit*), welches zum Testen eines Fehlers bei der Übertragung eines Zeichens dient. Es gibt Drucker, die GERADE Parität (das Paritätsbit wird so auf 0 oder 1 gesetzt, daß jeder Zeichencode eine gerade Zahl von Einsen enthält) oder UNGERADE Parität verwend (die Anzahl Einsen ist ungerade). Falls Ihr Drucker die Parität nicht überprüft, kann SPACE gewählt werden, wobei das achte Bit immer 0 ist, bzw. MARK, wobei das achte Bit immer 1 ist. Die Einstellung "NICHTS" ermöglicht die Übertragung aller acht Bits an den Drucker.

ZEILEN/SEITE and ZEICHEN/ZEILE bestimmen die maximale Anzahl Textzeilen (einschließlich der Leerzeilen bei doppeltem oder dreifachem Zeilenabstand) je Seite bzw. die maximale Anzahl Zeichen pro Zeile. Die in den Druckertreibern vorgesehenen Werte eignen sich für A4-Papier.

ENDLOSFORMULAR betrifft die Verwendung einer Papierrolle (JA) im Gegensatz zu einzelnen Blättern (NEIN). Beim Drucken mit einzelnen Blättern hält der Drucker am Ende jeder Seite an. Am Bildschirm macht Sie eine Meldung darauf aufmerksam, daß ein neues Blatt eingelegt werden muß. Zum Fortsetzen des Druckvorgangs dient **ENTER**, zum Abbrechen **ESC**.

Der ZEILENENDE-CODE ist die Codesequenz, die dem Drucker mitteilt, daß das Ende einer Zeile erreicht ist. Die Mehrzahl der Drucker akzeptiert einen Wagenrücklauf– gefolgt von einem Zeilenvorschubcode. Bei der Ausgabe eines SuperBASIC-Programms auf eine Datei empfiehlt es sich, als Markierung für das Zeilenende den Zeilenvorschub-Code zu wählen.

Die PRÄAMBEL– und POSTAMBEL-CODES sind bei manchen Druckern erforderlich, die vor der Inbetriebnahme eine Initialisierung benötigen. In diesem Zusammenhang können z.B. Randeinstellung gesetzt oder ein bestimmter Zeichensatz gewählt werden. Nach erfolgter Druckausgabe mit einem QL-Programm können diese Einstellungen wieder auf ihre Standardwerte zurückgesetzt werden. Dafür sind die Präambel– und Postambel-Parameter zuständig, die Ihnen erlauben, bis zu 10 Zeichen an den Drucker zu übermitteln.

FETTDRUCK EIN und FETTDRUCK AUS sind die Codes zum Ein– und Ausschalten der verstärkten Schrift zur Hervorhebung eines Texts. Falls Ihr Drucker nicht über diese Sonderfunktion verfügt, können Sie FEHLW benützen, vorausgesetzt Ihr Drucker reagiert auf den Rückschritt-Code.

UNTERSTREICHUNG EIN und AUS dient zum Ein– und Ausschalten der Unterstreichungsfunktion, unter der Voraussetzung, daß Sie einen Drucker mit automatischer Unterstreichung haben. Andernfalls können Sie FEHLW verwenden, allerdings nur, wenn der Drucker auf den Rückschritt-Code reagiert.

HOCHSTELLUNG/TIEFSTELLUNG EIN/AUS dienen zur Eingabe der Codesequenz, welche bei Ihrem Drucker Hoch– und Tiefstellung bewirkt und anschließend wieder auf die normale Textzeile zurückstellt.

Die Parameter ÜBERSETZEN 1 bis ÜBERSETZEN 10 akzeptieren bis zu 10 Zeichen, wobei das erste Zeichen vor der Übertragung an den Drucker in die nachfolgende Sequenz übersetzt wird. Dieses erste Zeichen darf kein Steuerzeichen sein, d.h. sein ASCII-Code muß im Bereich zwischen 32 und 255 liegen. Die Übersetzung kann beliebige Zeichen oder eine Folge von Steuercodes enthalten, in der Druckausgabe muß das Ergebnis ein einzelnes Zeichen sein.

	Standard	Andere Möglichkeiten
TREIBERNAME	: ANDERE	FX-80
ANSCHLUSS	: ser1	ser2
BAUD-RATE	: 9600	9600
PARITÄT	: NICHTS	
ZEILEN/SEITE	: 66	66
ZEICHEN/ZEILE	: 80	80
ENDLOSFORMULAR	: NEIN	JA
ZEILENENDE-CODE	: CR,LF	CR,LF
PRÄAMBLE-CODE	: NICHTS	ESC,@,ESC,R,NULL
POSTAMBLE-CODE	: NICHTS	
FETTDRUCK EIN	: NICHTS	ESC,E
FETTDRUCK AUS	: NICHTS	ESC,F
UNTERSTR. EIN	: NICHTS	ESC – 1
UNTERSTR. AUS	: NICHTS	ESC – 0
TIEFSTELLEN EIN	: NICHTS	ESC,S,1
TIEFSTELLEN AUS	: NICHTS	ESC,T
HOCHSTELLEN EIN	: NICHTS	ESC,S,0
HOCHSTELLEN AUS	: NICHTS	ESC,T
ÜBERSETZEN 1	: NICHTS	£,ESC,R,ETX,@,ESC,R,NULL
ÜBERSETZEN 2	: NICHTS	
ÜBERSETZEN 3	: NICHTS	
ÜBERSETZEN 4	: NICHTS	
ÜBERSETZEN 5	: NICHTS	
ÜBERSETZEN 6	: NICHTS	
ÜBERSETZEN 7	: NICHTS	
ÜBERSETZEN 8	: NICHTS	
ÜBERSETZEN 9	: NICHTS	
ÜBERSETZEN 10	: NICHTS	

Erstellen wir zur Übung einen zweiten Druckertreiber für den QL Printer. Laden und betreiben Sie das Installationsprogramm aus SuperBASIC und wählen Sie den Treiber "ANDERE", gefolgt von **F1** oder **F2**. Die nachstehende Zusammenstellung zeigt die Ausgangswerte und in der Spalte ganz rechts die Werte für den QL Printer.

Ändern Sie zuerst den Druckernamen, indem Sie "ANDERE" mit dem Rechtspfeil löschen und eingeben:

```
QL Printer [ENTER]
```

Falls Ihnen dabei ein Fehler passiert, wiederholen Sie das Ganze nochmals.

Drücken Sie die Abwärtspfeiltaste, bis der Parameter ENDLOSPAPIER hervorgehoben wird. Hier gibt es nur zwei Optionen. Durch Drücken des Rechts- oder des Linkspfeils schalten Sie von NEIN auf JA.

Als Präambel für den QL Printer eignet sich **CAN**, welches den Druckerpuffer leert.

Wählen Sie mit der Abwärtspfeiltaste den PRÄAMBEL-Code und löschen Sie den vorgegebenen Wert mit dem Links- oder Rechtspfeil.

Die Präambel-Sequenz kann zusätzliche Druckerfunktionen festlegen, beispielsweise Zeilenabstand oder Schrägschrift. Wenn Ihr Drucker ohne Initialisierung auskommt, lassen Sie die Einstellung auf "NICHTS".

Der QL Printer erfordert als Postambel-Code ein **LF** oder **FF**. Folglich belassen wir die Einstellung unverändert.

FETTDRUCK EIN und FETTDRUCK AUS erfordern auf dem QL Printer die Codes **ESC E** und **ESC F**, welche Sie wie folgt eingeben können:

```
esc,"E
esc,"F
```

Die restlichen Codes werden so eingegeben:

Parameter	Eingabe
UNTERSTR. EIN	`esc,"-,"1`
AUS	`esc,"-,"0`
TIEFSTELLEN EIN	`esc,"S,"1`
AUS	`esc,"T`
HOCHSTELLEN EIN	`esc,"S,"0`
AUS	`esc,"T`

Wenn Sie alle notwendigen Änderungen an den Druckercodes vorgenommen haben, können Sie den neuen Drucker durch Drücken von **F5** installieren oder mit **ESC** zur Liste der Drucker zurückkehren.

PARALLELE DRUCKER

Mit Hilfe des Installationsprogramms ist es auch möglich, den QL mit einem Drucker zu betreiben, der nicht über einen der seriellen Anschlüsse **ser1** oder **ser2** verbunden wird. Davon wird Gebrauch gemacht, wenn eine parallele Schnittstelle angebracht wurde. Laden und betreiben Sie **install_bas** gemäß der vorherigen Beschreibung. Nach Wahl der Installation auf Microdrive 1 oder 2 drücken Sie die Leertaste für die Option "Parallel-Anschluß".

Auch hier wird Ihnen die Liste mit den Druckern angezeigt, doch nach Drücken von **F1** bzw. **F2** präsentiert Ihnen das Programm die folgenden Parameterwerte:

	Standard	Mögliche Optionen
TREIBERNAME	: STANDARD	....
ANSCHLUSS	: Keine	....
ZEILEN/SEITE	: 66	0 bis 255
ZEICHEN/ZEILE	: 80	0 bis 255
ENDLOSFORMULARE	: JA	NEIN
ZEILENENDE-CODE	: CR,LF	....
PRÄAMBLE-CODE	: NICHTS	....
POSTAMBLE-CODE	: NICHTS	....
FETTDRUCK EIN	: NICHTS	....
FETTDRUCK AUS	: NICHTS	....
UNTERSTR. EIN	: NICHTS	....
UNTERSTR. AUS	: NICHTS	....
TIEFSTELLEN EIN	: NICHTS	....
TIEFSTELLEN AUS	: NICHTS	....
HOCHSTELLEN EIN	: NICHTS	....
HOCHSTELLEN AUS	: NICHTS	....
ÜBERSETZEN 1	: NICHTS	....
ÜBERSETZEN 2	: NICHTS	....
ÜBERSETZEN 3	: NICHTS	....
ÜBERSETZEN 4	: NICHTS	....
ÜBERSETZEN 5	: NICHTS	....
ÜBERSETZEN 6	: NICHTS	....
ÜBERSETZEN 7	: NICHTS	....
ÜBERSETZEN 8	: NICHTS	....
ÜBERSETZEN 9	: NICHTS	....
ÜBERSETZEN 10	: NICHTS	....

Die beiden Optionen Baud-Rate und Parität stehen hier nicht zur Auswahl, weil sie lediglich für die seriellen Anschlüsse **ser1** oder **ser2** von Bedeutung sind. Der einzige andere Unterschied gegenüber der Installation eines seriellen Druckers ist beim Parameter ANSCHLUSS. Dieser ist mit dem Rechts– oder Linkspfeil zu löschen und durch einen beliebigen zulässigen *Einheitennamen* mit bis zu 16 Zeichen zu ersetzen (in diesem Zusammenhang sei auf den Abschnitt *Begriffe* bzw. auf die Begleitdokumentation zu dem entsprechenden Peripheriegerät verwiesen).

Ansonsten unterscheidet sich die Installation eines parallelen Druckers nicht von der eines seriellen.

DAS QL PROGRAM
CONFIG_BAS

Das Programm **config_bas** erlaubt Ihnen, alternative Standardeinheiten zum Gebrauch durch die QL Programme zu spezifizieren und die Sortierreihenfolge der **Ordnen**-Befehle in Abacus und Archive zu ändern. Sie finden es auf der Kassette Utilities.

Die Originalprogramme verwenden standardmäßig Microdrive 2 zum Speichern von Daten und erwarten, daß der installierte Druckertreiber und die Hilfe-Datei auf Microdrive 1 vorliegen. Falls Sie zusätzliche Microdrives, Laufwerke usw. verwenden wollen, können entsprechende Vorkehrungen getroffen werden.

Bei Bedarf kann auch die Sortierreihenfolge bei Archive– Datensätzen und in Abacus-Kalkulationstabellen geändert werden.

EINSATZ VON CONFIG

Config_bas kann von einem beliebigen Microdrive aus betrieben werden, um ein QL-Programm in Microdrive 1 oder 2 zu modifizieren. Der Befehl zum Betreiben von **config_bas** in Microdrive 2, zum Modifizieren einer Reservekopie eines QL-Programms in Microdrive 1 lautet:

```
lrun mdv2_config_bas
```

Auf Anfrage geben Sie den Namen des Programms ein, an dem die Änderung vorgenommen werden soll (Quill, Abacus, Easel oder Archive), und drücken **ENTER**. Die Frage nach dem Microdrive beantworten Sie mit 1.

Das **config_bas**-Programm wartet, bis Sie sich vergewissert haben, daß die Kassetten in den entsprechenden Microdrives liegen, und **ENTER** drücken. Danach zeigt es Ihnen das Hauptmenü mit den folgenden Optionen:

Neue Standardgeräte wählen
Sortierreihenfolge ändern
Programm verlassen

Sortierreihenfolge

Die Option zum Ändern der Sortierreihenfolge mit **ENTER** aktivieren und bei Aufforderung die Leertaste drücken.

Der größte Teil des Bildschirms wird von einem Block mit 256 Zeichen eingenommen, deren Plätze die Sortierreihenfolge definieren, wobei das zu sortierende Zeichen durch die *Cursor-Position* innerhalb des Blocks (von links nach rechts und von oben nach unten) bestimmt wird. Der jeweilige *Inhalt* zeigt, wie das betreffende Zeichen durch den **Ordnen**-Befehl getestet wird. Rechts auf dem Bildschirm werden weitere Informationen über das Zeichen an der Cursorposition angezeigt. Die Steuerung des Cursors erfolgt mit den Pfeiltasten.

Der Zeichenblock im unteren Bildschirmbereich dient zur Modifikation der Reihenfolge. Hier gibt es einen separaten Cursor, den Sie mit **SHIFT** und den Pfeiltasten bewegen. Wie Sie sehen, enthält dieser Block nur die eine Hälfte des Zeichensatzes. **F1** schaltet zwischen den beiden Hälften hin und her.

Wir beschreiben in zwei Beispielen, wie Sie die Sortierreihenfolge verändern können. Die eingebaute Sortierfunktion ordnet die Kleinbuchstaben bei den Großbuchstaben ein, das heißt, sie unterscheidet nicht zwischen Groß und Kleinschreibung. Nehmen wir an, Sie wollen eine Sortierung, bei der die Kleinbuchstaben nach den Großbuchstaben eingeordnet werden, bei der also "a" nach "Z" sortiert wird.

Im Hauptblock sehen Sie die Großbuchstaben zweimal. Der zweite Block von Großbuchstaben steht an den Plätzen, auf die eigentlich die Kleinbuchstaben hingehören. Plazieren Sie jetzt den Cursor im Hauptblock auf das Zweite "A". Rechts auf dem Bildschirm sehen Sie, daß dies der Platz von "a" ist. Geben Sie jetzt ein kleines "a" ein. Im rechten Teil des Bildschirms steht jetzt die Information "a wird sortiert wie a", das heißt, es wird bei den Kleinbuchstaben im Anschluß an die Großbuchstaben sortiert. Verfahren Sie so bei allen anderen Kleinbuchstaben.

Eine alternative Methode zum Ändern von Buchstaben ist durch Bewegen des Cursors im unteren Zeichenblock (mit **SHIFT** und den Pfeiltasten) auf das gewünschte Zeichen und Drücken von **F2**. Diese Methode bietet sich vor allem an, wenn es um Zeichen geht, die auf der Tastatatur nicht vorgesehen sind. Im folgenden ein Beispiel zur Veranschaulichung.

Nehmen wir an, die übliche Sortierreihenfolge für Großbuchstaben soll umgekehrt werden, alles andere jedoch unveränder bleiben. Zu diesem Zweck muß der betreffende Teil im Hauptblock, also "A B C ... X Y Z", auf "Z Y X ... C B A" geändert werden. Setzen Sie den Cursor im Hauptblock auf "A" und den unteren Cursor auf "Z". Drücken Sie

dann **F2** zur Eingabe des neuen Zeichens. Das "A" wird jetzt bei der Sortierung wie ein "Z" behandelt. Wiederholen Sie diesen Vorgang für den Rest des Alphabets, indem Sie "B" durch "Y", "C" durch "X" ersetzen, usw.

Nach Festlegung der neuen Sortierordnung speichern Sie sie mit **F5** auf der Kassette ab, wo sie die alte ersetzt. Mit **ESC** kehren Sie zum Hauptmenü zurück.

Einheiten

Standardmäßig verwenden die QL Programme Microdrive 1 für Systeminformationen (z.B. zum Lesen des eingebauten Druckertreibers) und für die HILFE-Funktion, während Microdrive 2 für Datenspeicherung vorgesehen ist.

Aus dem Hauptmenü wählen Sie mit der Leertaste die Option "Einheiten ändern" an. Drücken Sie wiederum die Leertaste, wenn das Programm Sie dazu auffordert.

Nach dem Einlesen der aktuellen Einstellung von der Kassette zeigt Ihnen das Programm die Werte und wartet auf Ihre Neueingabe. Zum Beibehalten der Standardwerte drücken Sie einfach **ENTER**, zum Ändern überschreiben Sie sie und bestätigen die Eingabe mit **ENTER**.

Anschließend können Sie die neuen Einstellungen speichern, weitere Änderungen vornehmen oder die Option abbrechen und zum Hauptmenü zurückkehren.

Wenn Sie die neuen Einstellungen abspeichern, gelten diese so lange, bis sie mit **config_bas** erneut geändert werden.

ARITHMETIK

Der zulässige Zahlenbereich für die QL– Programme, mit Ausnahme von QL Easel, ist:

$\pm 2.9*10^{-39}$ to $\pm 1.7*10^{38}$

Alle Berechnungen werden mit einer Genauigkeit von 16 Stellen durchgeführt, wobei höchstens 14 Stellen zur Anzeige gelangen.

Der Zahlenbereich in Easel ist:

$\pm 1.0*10^{-35}$ to $\pm 1.0 \times 10^{36}$

Die folgenden arithmetischen Operatoren sind in in Abacus, Archive und Easel verfügbar:

Operator	Funktion
+	Addition von Zahlen und Verknüpfung von Zeichenketten
–	Subtraktion von Zahlen
*	Multiplikation
/	Division
^	Potenzieren
=	Gleichheit
>	Größer als
<	Kleiner als
< =	Kleiner oder gleich
> =	Größer oder gleich
< >	Ungleichheit

Es erfolgt keine automatische Datentypumwandlung. Folglich müssen die Operanden vom gleichen Typ sein. Bei bedingten Ausdrücken ist das Ergebnis in jedem Fall eine Zahl; 1 wenn die Bedingung wahr ist, andernfalls 0.

Funktionen und Operatoren weisen die folgende hierarchische Rangfolge auf:

Operation	Prioritätsebene
Indizieren und Abtrennen	12
Alle Funktionen	11
^	10
Unäres Minus	9
*,/	8
+.–	6
=,>,<,< =,> =,< >	5
Nicht (NOT)	4
Und (AND)	3
Oder (OR)	2

FORMATIEREN

Beim Formatieren einer Kassette werden alle dort gespeicherten Daten überschrieben und gehen unwiederbringlich verloren. Vergewissern Sie sich daher unbedingt, daß Sie nur fabrikneue Kassetten formatieren bzw. solche, die keine wertvollen Informationen enthalten.

Denken Sie sich einen Namen für die Kassette aus, der nicht länger als zehn Zeichen sein darf. Achten Sie darauf, daß der QL eingeschaltet und der blinkende Cursor sichtbar ist, und legen Sie die Kassette in den Microdrive 1. Nennen wir unsere Kassette "Daten". Der erforderliche Befehl zum Formatieren lautet dann:

```
FORMAT mdv1_daten
```

Verwechseln Sie das Unterstreichungszeichen (__) nicht etwa mit dem Minuszeichen (–), welches sich auf derselben Taste befindet. Das Unterstreichungszeichen ist das Zeichen in der oberen Umschaltstellung, d.h. Sie müssen gleichzeitig **SHIFT** festhalten.

Drücken Sie **ENTER**, worauf das Lämpchen des linken Microdrive etwa 30 Sekunden lang leuchtet. Der QL gibt eine Nachricht über den verfügbaren Speicherplatz auf der Kassette aus. Eine ausführliche Beschreibung zum Formatierungsbefehl finden Sie im Abschnitt Befehle.

Es ist angezeigt, eine fabrikneue Kassette mehrere Male zu formatieren. Auf diese Weise läuft das Band besser und die Speicherkapazität wird erhöht.

Natürlich können Sie eine Kassette auch in Microdrive 2 formatieren, indem Sie **mdv2__** statt **mdv1__** eingeben.

RESERVEKOPIEN

Unter einer Reservekopie versteht man eine exakte Kopie aller Dateien auf einer Kassette auf eine zweite, leere Kassette. Es empfiehlt sich, dazu eine neu formatierte Kassette zu verwenden und ihr einen Namen zu geben, der anzeigt, daß es sich um eine Zweitkopie einer Originalkassette handelt.

Nehmen Sie eine leere Kassette oder eine, deren Daten Sie nicht mehr benötigen, und legen Sie sie in Microdrive 1. Geben Sie ihr einen Namen, etwa "daten__res" für eine Reservekopie der Originalkassette "daten". Schreiben Sie dann:

```
FORMAT mdv1_daten_res
```

gefolgt von **ENTER**. Darauf leuchtet das Lämpchen des linken Microdrive etwa 30 Sekunden lang.

Legen Sie die Originalkassette in Microdrive 2 und geben Sie ein:

```
DIR mdv2_
```

worauf am Bildschirm eine Liste der Dateien ausgegeben wird.

Geben Sie für jede aufgelistete Datei einen Befehl dieser Art ein:

`COPY mdv2_` *Dateiname* `TO mdv1_` *Dateiname*

wobei jeweils für "Dateiname" der betreffende Name einzusetzen ist. Dieser Befehl kopiert jede benannte Datei aus Microdrive 2 auf Microdrive 1. Je nach dem Umfang der einzelnen Dateien dauert dieser Vorgang unterschiedlich lang und kann einige Zeit in Anspruch nehmen.

Wiederholen Sie den **COPY** Befehl für alle aufgelisteten Dateien. Nach Abschluß des Kopiervorgangs ist die Reservekopie aus Microdrive 2 mit dem Namen und mit einem Vermerk über die Originalkassette zu versehen und dann an einem sicheren Ort zu verwahren.

Es empfiehlt sich, von jeder Kassette, mit der Sie arbeiten, eine, zwei oder mehr Reservekopien zu erstellen, je nachdem, wie wertvoll die darauf befindlichen Informationen sind. Verwenden Sie zum Erstellen der Reservekopie jeweils die Kassette mit der jüngsten Version der Dateien.

Dezimal	Hex	Tasten	Anzeige/Funktion
48	30	0	0
49	31	1	1
50	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	SHIFT .	:
59	3B	SHIFT ,	;
60	3C	<	<
61	3D	SHIFT 0	=
62	3E	SHIFT <	>
63	3F	SHIFT ß	?
64	40	CTRL \	@
65	41	SHIFT A	A
66	42	SHIFT B	B
67	43	SHIFT C	C
68	44	SHIFT D	D
69	45	SHIFT E	E
70	46	SHIFT F	F
71	47	SHIFT G	G
72	48	SHIFT H	H
73	49	SHIFT I	I
74	4A	SHIFT J	J
75	4B	SHIFT K	K
76	4C	SHIFT L	L
77	4D	SHIFT M	M
78	4E	SHIFT N	N
79	4F	SHIFT O	O
80	50	SHIFT P	P
81	51	SHIFT Q	Q
82	52	SHIFT R	R
83	53	SHIFT S	S
84	54	SHIFT T	T
85	55	SHIFT U	U
86	56	SHIFT V	V
87	57	SHIFT W	W
88	58	SHIFT X	X
89	59	SHIFT Y	Y
90	5A	SHIFT Z	Z
91	5B	CTRL 9	[
92	5C		
93	5D	CTRL 0	]
94	5E	SHIFT \	ˆ
95	5F	SHIFT –	___
96	60	CTRL 7	£
97	61	A	a
98	62	B	b
99	63	C	c
100	64	D	d
101	65	E	e
102	66	F	f
103	67	G	g
104	68	H	h
105	69	I	i
106	6A	J	j
107	6B	K	k
108	6C	L	l
109	6D	M	m
110	6E	N	n
111	6F	O	o

Dezimal	Hex	Tasten	Anzeige/Funktion
112	70	P	p
113	71	Q	q
114	72	R	r
115	73	S	s
116	74	T	t
117	75	U	u
118	76	V	v
119	77	W	w
120	78	X	x
121	79	Y	y
122	7A	Z	z
123	7B	CTRL ß	{
124	7C	CTRL 8	|
125	7D	CTRL #	}
126	7E	CTRL <	~
127	7F	SHIFT ESC	©
128	80	Ä	ä
129	81	CTRL SHIFT 1	ã
130	82	CTRL SHIFT Ä	å
131	83	CTRL SHIFT 3	é
132	84	Ö	ö
133	85	CTRL SHIFT 5	õ
134	86	CTRL SHIFT 7	ø
135	87	Ü	ü
136	88	CTRL SHIFT 9	ç
137	89	CTRL SHIFT 0	ñ
138	8A	CTRL SHIFT 8	æ
139	8B	CTRL SHIFT #	œ
140	8C	CTRL ,	á
141	8D	CTRL SHIFT 4	à
142	8E	CTRL .	â
143	8F	CTRL –	ë
144	90	CTRL SHIFT V	è
145	91	CTRL 1	ê
146	92	CTRL 2	ï
147	93	CTRL 3	í
148	94	CTRL 4	ì
149	95	CTRL 5	î
150	96	CTRL 6	ó
151	97	CTRL SHIFT ,	ò
152	98	CTRL SHIFT D	ô
153	99	CTRL Ä	ú
154	9A	CTRL SHIFT Ö	ù
155	9B	CTRL Ö	û
156	9C	ß	ß
157	9D	CTRL SHIFT G	¢
158	9E	CTRL SHIFT .	¥
159	9F	CTRL SHIFT –	'
160	A0	SHIFT Ä	Ä
161	A1	CTRL SHIFT A	Ã
162	A2	CTRL SHIFT B	Å
163	A3	CTRL SHIFT C	É
164	A4	SHIFT Ö	Ö
165	A5	CTRL SHIFT E	Õ
166	A6	CTRL SHIFT F	Ø
167	A7	SHIFT Ü	Ü
168	A8	CTRL SHIFT H	Ç
169	A9	CTRL SHIFT I	Ñ
170	AA	CTRL SHIFT J	Æ
171	AB	CTRL SHIFT K	Œ
172	AC	CTRL SHIFT L	α
173	AD	CTRL SHIFT M	δ
174	AE	CTRL SHIFT N	θ
175	AF	CTRL SHIFT O	λ

Dezimal	Hex	Tasten	Anzeige/Funktion
176	B0	CTRL SHIFT P	μ
177	B1	CTRL SHIFT Q	π
178	B2	CTRL SHIFT R	Φ
179	B3	CTRL SHIFT S	¡
180	B4	CTRL SHIFT T	¿
181	B5	CTRL SHIFT U	[illegible]
182	B6	SHIFT 3	§
183	B7	CTRL SHIFT W	¤
184	B8	CTRL SHIFT X	≪
185	B9	CTRL SHIFT z	≫
186	BA	CTRL SHIFT Y	°
187	BB	CTRL Ü	÷
188	BC	CTRL SHIFT 2	←
189	BD	CTRL +	→
190	BE	CTRL SHIFT 6	↑
191	BF	CTRL SHIFT ß	↓
192	C0	←	Cursor ein Zeichen nach links
193	C1	ALT ←	Cursor an Anfang der Zeile
194	C2	CTRL ←	ein Zeichen links löschen
195	C3	CTRL ALT ←	Zeile löschen
196	C4	SHIFT ←	Cursor ein Wort nach links
197	C5	SHIFT ALT ←	PAN links
198	C6	SHIFT CTRL ←	ein Wort links löschen
199	C7	SHIFT CTRL ALT ←	
200	C8	→	Cursor ein Zeichen nach links
201	C9	ALT →	Cursor an das Ende der Zeile
202	CA	CTRL →	Zeichen unter Cursor löschen
203	CB	CTRL ALT →	bis Zeilenende löschen
204	CC	SHIFT →	Cursor ein Wort rechts
205	CD	SHIFT ALT →	PAN rechts
206	CE	SHIFT CTRL →	Wort unter und rechts von Cursor löschen
207	CF	SHIFT CTRL ALT →	
208	D0	↑	Cursor hoch
209	D1	ALT ↑	Rollen hoch
210	D2	CTRL ↑	Suchen rückwärts
211	D3	CTRL ALT ↑	
212	D4	SHIFT ↑	
213	D5	SHIFT ALT ↑	
214	D6	SHIFT CTRL ↑	
215	D7	SHIFT CTRL ALT ↑	
216	D8	↓	Cursor unten
217	D9	ALT ↓	Rollen unten
218	DA	CTRL ↓	Suchen vorwärts
219	DB	CTRL ALT ↓	
220	DC	SHIFT ↓	Ende von Bildschirm
221	DD	SHIFT ALT ↓	
222	DE	SHIFT CTRL ↓	
223	DF	SHIFT CTRL ALT ↓	
224	E0	↕	Umschalten Ein/Aus
225	E1	ALT	
226	E2	CTRL	
227	E3	ALT CTRL	
228	E4	SHIFT	
229	E5	SHIFT ALT	
230	E6	SHIFT CTRL	
231	E7	SHIFT CTRL ALT	
232	E8	F1	
233	E9	CTRL F1	
234	EA	SHIFT F1	
235	EB	CTRL SHIFT F1	
236	EC	F2	
237	ED	CTRL F2	
238	EE	SHIFT F2	
239	EF	CTRL SHIFT F2	

Dezimal	Hex	Tasten	Anzeige/Funktion
240	F0	F3	
241	F1	CTRL F3	
242	F2	SHIFT F3	
243	F3	CTRL SHIFT F3	
244	F4	F4	
245	F5	CTRL F4	
246	F6	SHIFT F4	
247	F7	CTRL SHIFT F4	
248	F8	F5	
249	F9	CTRL F5	
250	FA	SHIFT F5	
251	FB	CTRL SHIFT F5	
252	FC	SHIFT Leerzeichen	"spezielles" Leerzeichen
253	FD	SHIFT TAB	TAB zurück (CTRL ignoriert)
254	FE	SHIFT ENTER	"spezielles" ENTER (CTRL ignoriert)
255	FF	Siehe unten	

Wenn die ALT-Taste mit einer beliebigen anderen Taste, außer den Cursor-Steuertasten oder der UMSCHALT-Taste betätigt wird, wird der Code FF erzeugt, gefolgt von einem Byte, das anzeigt, was der Tastencode gewesen wäre, wenn die ALT-Taste nicht gedrückt wäre.

Zeichen bis 20 Hex sind Kontrollzeichen oder nicht druckbare Zeichen. Alternative Zeichen sind in Klammern gesetzt.

CTRL-C wird von Qdos abgefangen und kann nicht erkannt werden ohne Änderung der System Variablen.

Die Zeichen zwischen C0 und DF sind Cursor-Kontroll Kommandos.

Die UMSCHALT-Taste (Caps Lock) und CTRL-F5 werden von Qdos abgefangen und können nur mit spezieller Software erkannt werden.

Sinclair QL Preservation Project (SQPP)



On January 12th 1984 Sir Clive Sinclair presented the Sinclair QL Professional Computer in a typical Sinclair-extravaganza type launch event at the Intercontinental Hotel, Hyde Park Corner, London. This was exactly 12 days earlier than Steve Jobs presented the Apple Macintosh.

The QL still is a very good example of an innovative, stylish, powerful and underestimated product. On one hand it failed in the market but on the other hand it influenced many developments which ended in many of today's computers.

2009 was the year of its 25th anniversary in which month by month new activities were launched.



Jan 12th – 25th Launch anniversary day. Message spread to VIP, community and media.
http://tinyurl.com/ql-is-25



Feb 19th – Massive 11 pages coverage of the QL in the April Issue of Personal Computer World (PCW) magazine.
http://www.pcw.co.uk



Mar 12th – Sinclair QL Preservation Project (SQPP) launched, starting with Documents/Publications from Sinclair Research Ltd and various computer magazines of the years 1984 to 1986.
http://tinyurl.com/sqpp25

2014 is the year of the 30th anniversary. Check out the brand new website http://www.qlis30.org.uk/. Activities include THE MOVIE, THE STORY, THE REPOSITORY, THE DISTRIBUTION and much more to come. Stay tuned...

QL forever!

Urs König (aka QLvsJAGUAR)
http://www.qlvsjaguar.homepage.bluewin.ch
http://www.youtube.com/QLvsJAGUAR
https://plus.google.com/104042128125238901905/posts